TERRY KEPNER'S

$3.95/CAN $4.95

portable 100

TANDY LAPTOP COMPUTING VOLUME 5, NUMBER 4 APRIL 1988

Composing With The Tandy 100

Why Your Tandy 100 Loses Memory

Delightful Data Entry

Using Shells In Basic

Peptone's Dynamic Duo

Tandy Portables Flunk Math

Announcing:

The SOUNDSIGHT GOLD CARD

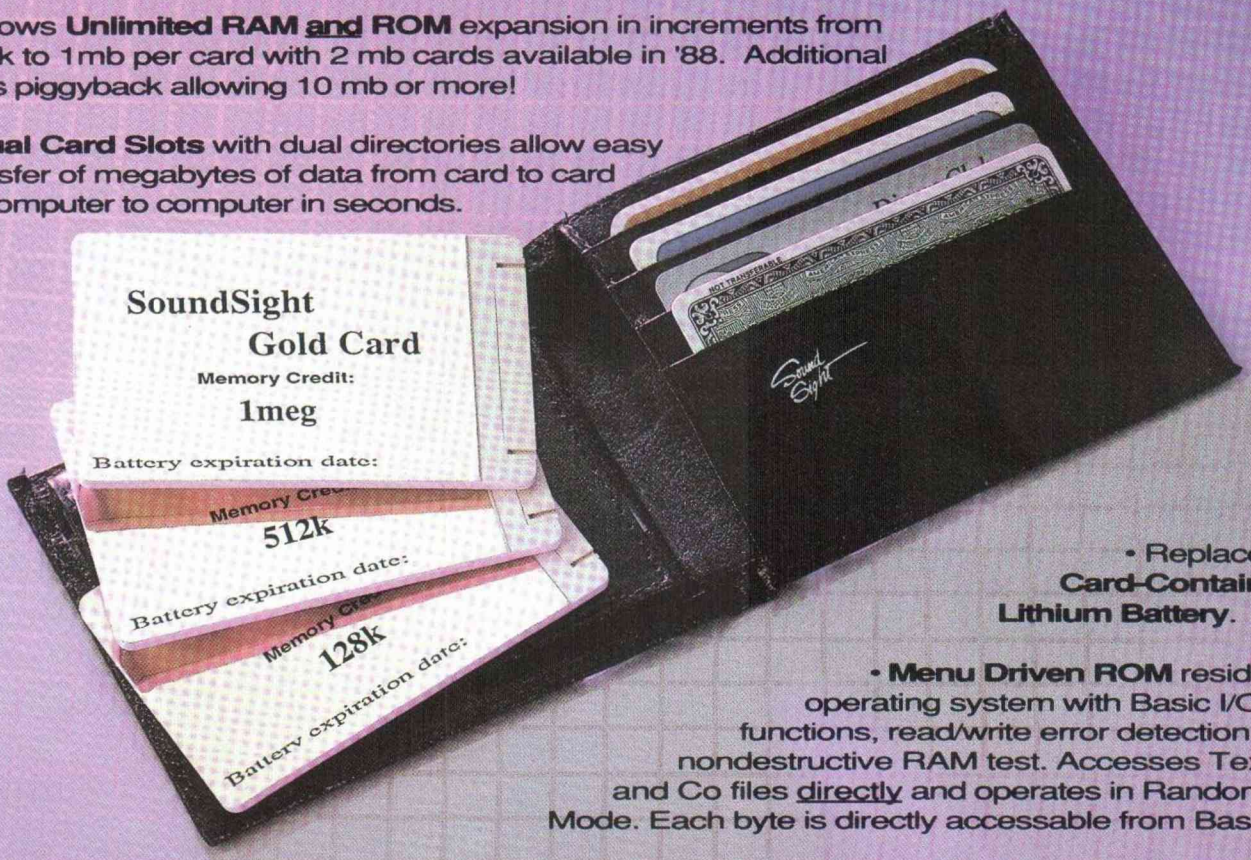
Megabytes of memory on a credit-card-sized wafer

A mere 1/8" thick

*Give your **Tandy 100/102** the power it deserves without sacrificing portability or adding bulky boxes. The SoundSight Gold Card lets you add as many megabytes of **RAM and ROM** as you wish with advantages other expansions just can't match.*

• Allows **Unlimited RAM and ROM** expansion in increments from 128k to 1mb per card with 2 mb cards available in '88. Additional slots piggyback allowing 10 mb or more!

• **Dual Card Slots** with dual directories allow easy transfer of megabytes of data from card to card or computer to computer in seconds.



• Replaceable, **Card-Contained Lithium Battery**.

• **Menu Driven ROM** resident operating system with Basic I/O functions, read/write error detection, and nondestructive RAM test. Accesses Text, Basic and Co files directly and operates in Random Access Mode. Each byte is directly accessable from Basic.

• SoundSight Gold Cards are also available in permanent **One-Time-Programmable or Read-Only-Reprogrammable** memory cards for large nonvolatile software application use in conjuction with RAM cards.

Optional new **ROM ELIMINATOR** lets you load and run multiple ROMs directly from the SoundSight Gold Card, dispensing with the need for extra ROM sockets or even having your ROM chips with the computer!

NEW OPTIONAL SOFTWARE DEVELOPMENTS:

- ROM resident **Forms Generator** with full Telecommunications and Desk Top package for Law Enforcement Incident Reports, or any custom data entry applications.
- Point of Sale **Inventory Control Software**
- New **Basic Compiler** compiles basic programs to machine code, condensing them to fraction of there original size and speed. Runs them directly from cards. *(Avail. Feb. '88.)*
- New **Text Editor** opens and edits files as large as available ram. *(Avail. Feb. '88.)*

*SoundSight offers **customized hardware/software** systems for law enforcement, retail, warehousing/distribution and other vertical markets, as well as third party dev. (C language).*
We're ready to meet your needs TODAY!

Pricing available from **128k** to **10mb** with **128k at $399.95**; **256k at $549.95**; and **512k at $749.95**; Additional cards for backup are available.

CONTACT:

SoundSight MBM Inc., 225 West Broadway, Suite #509, Glendale, CA 91204
Phone (213) 463-9457 or (818) 240-8400

FREE ISSUE

Add POWER to your portable with PICO—The Magazine of PORTABLE COMPUTING !

Take your first issue FREE with the coupon below!

"Portable, Briefcase, Laptop, Notebook."

Whatever you call them, PICO sizes them all up every month. PICO is where you'll find the hardware, the software, the applications, peripherals and more that help you get **all** the power your portable can deliver!

Issue after issue, PICO's computing pros bring you valuable how-to's, tips and techniques, extended tutorials, on-target news, reviews and previews, software updates, buyer's guide – all designed to make you a better computerist, whatever your favorite portable brand or flavor.

"The hottest magazine for the hottest new technology in computing!" says tech consultant Greg Resker.

SEE FOR YOURSELF WITH YOUR FIRST ISSUE, FREE!

Complete the coupon and mail today to get your first issue of PICO FREE and without obligation of any kind. If you're not satisfied, just write "cancel" on the invoice and owe nothing. The issue is yours to keep. Otherwise, a full year of PICO's money-saving help and expertise is just $29.97, GUARANTEED. Cancel at any time and receive prompt refund of the unused portion of your subscription.

SPECIAL **BONUS**

Send payment with your order – with full refund guarantee – and get PICO's exclusive Buyer's Guide to 88 current portables including data on 17 performance catagories!

Subscription includes FREE MEMBERSHIP on *The Source* database (a $49.95 value) plus access to the PICO BBS!

Buy smart – Computer smart with PICO.
Complete the coupon now and mail today!

Mail to: PICO P.O. Box 428, Peterborough, NH 03458-0428

YES! send my first issue of PICO to examine for 30 days, FREE!
Also enter my no-risk subscription for one year at $29.97* – 30% off the cover price.
Subscription includes membership in THE SOURCE, a $49.95 value, plus access to the PICO BBS. If not satisfied, I'll mark "cancel" on the invoice and owe nothing.

❑ Payment/Charge enclosed. Please send BONUS
❑ Bill me. I'll skip the BONUS—Portable Computer Buyer's Guide.
❑ 2 Yrs, $55 (**Save $5**) (Full refund guarantee.)
❑ CHARGE MY: ❑ MC ❑ Visa Card# _____ Exp. Dt. _____

(print) NAME _____

Address _____

City _____ State _____ Zip _____

Computer Type _____

*Canada & Mexico, $32.97, all other foreign, $49.97. Airmail add $50.00, US funds drawn on US banks only. Please allow 4-6 weeks for delivery of your first issue.

ON THE COVER:
Composing with a Tandy 102 (keyboard courtesy of Radio Shack)

Photo by:
Diana Shonk Wallace

VOL. 5, NO. 4
APRIL 1988

Tandy 100
Tandy 200
Tandy 600

ROM WITH A VIEW

This month we tune in on a very innovative use for the Notebook marvel... music. Author John Comeau has been busy making some minor adjustments to the cabling to make the Model 100 a perfect companion for his electronic piano... somewhat "techie" but very worthwhile.

This article brings to light one of the most fascinating aspects of the Model 100: versatility. Everything the notebook computer touches seems to be improved or enhanced. In the past, we've seen the model 100 improve telephones, calculators, cash registers, cellular phones, mainframe computers, children's toys, and more. In the future, I expect we'll see notebook computers interfaced with hundreds of other household and office devices. I've heard that some major appliance companies are connecting dishwashers and refrigerators to RS-232 outputs for diagnostics (they call for service themselves). Programmable microwaves, computer-connected VCR's, programmable vacuum cleaners and lawnmowers... what's next?

The way we publish this magazine is an example of technology utilized to the maximum. Authors around the country type in articles and reviews on their notebook (and sometimes other) computers. Often, they'll call our bulletin board system and upload the article to us, but usually they send us their article on tape or disk. Then it's transferred to our proofreading computer, usually an MS-DOS portable like the Wang or Datavue Spark. The article gets edited in MicroSoft Word. From there, it's transferred to a Tandy 4000 computer... a real winner... and Aldus PageMaker where it's roughly designed into the neat little columns and boxes which you see. The next step is Diana's Macintosh where it is final-designed into an article which is not only informative, but looks appealing, too. Once complete, the diskette is wisked off to a National Colorite's Linotronic 300 typesetting machine which converts the PageMaker page images to negatives... ready to be printed. The characters you are reading now are direct from my computer, never having seen paper prior to the actual magazine pages.

This is our fourth issue of Portable 100 since we purchased it from C.W. Communications. It took us some time to get it up and running, but now that the system is in place, we're ready to make some major changes to the look and feel of the magazine. Please watch for them... we're quite proud of PORTABLE 100.

Would you like to help us out? If you have a moment, check your address label. There should be a six digit number which corresponds to your expiration date, for example 890328 designates an expiration date of March 28, 1989. If our records don't match yours, send off a note to us. If your subscription is about to expire, send us a renewal... you'll be saving us some serious time and money and saving yourself the agony of possibly missing a valuable issue.

Just a side-note: Our former fulfillment house, who has since been fired, has been getting a number of calls from people who want to subscribe. Their 800 number is on some of our old mailings and literature. We recently found out that they have been saying "I'm Sorry, but Portable 100 is no longer in business". Well guys... APRIL FOOLS! We're still here and better than ever!

- Mark Robinson

PRESIDENT/PUBLISHER
Terry Kepner

EXECUTIVE VICE PRESIDENT
Mark Robinson

EDITOR-IN-CHIEF
Terry Kepner

EDITORS
JoAnn Niemela, Linda Tiernan

CIRCULATION MANAGER
David Wallace

ADVERTISING DIRECTOR
Randy Byers

TECHNICAL CONSULTANT
Gregory F. Resker

DESKTOP PUBLISHING
ACROSS THE BOARD
Graphic Design, Inc.

ART DIRECTOR
Diana Shonk Wallace

PORTABLE COMPUTING INTERNATIONAL CORPORATION
145 Grove St. Ext., #21, PO Box 428
Peterborough, NH 03458-0428

Editorial 603-924-7859

Advertising 603-924-7949

Circulation 603-924-7949

Portable 100 (ISSN 0893-942X) is published by Portable Computing International Corporation, 145 Grove Street Ext., #21, P.O. Box 428, Peterborough, NH 03458-0428. *Portable 100* is an independent journal not connected with any hardware, software, or peripheral equipment manufacturer. *Portable 100* is published monthly, except for a combined July/August issue in the summer.

Subscription Service: All subscription correspondence should be addressed to *Portable 100*, Portable Computing International Corporation, 145 Grove Street Ext. #21, P.O. Box 428, Peterborough, NH 03458-0428. U.S. subscription rates: $24.97, one year; $53 two years. Canada and Mexico: $29.97, one year; $61 two years. All other foreign (surface mail): $44.97, one year; $85 two years. Foreign Air Mail, add $50 per subscription year. All payment U.S. funds drawn on U.S. Bank. Second-class postage paid at Peterborough, NH 03458, and at additional mailing offices.

POSTMASTER: Send address changes to:
Portable 100, Portable Computing International Corporation, 145 Grove Street Ext., #21, P.O. Box 428, Peterborough, NH 03458-0428.

Cruisin' Corrections

I am writing in regards to the article *Cruisin with the Model 100* in the January 1988 issue. There are some basic design flaws with the power supply shown. While it is true you can get approximately six volts by tricking a five volt regulator with the resistor divider, it is not necessary as you can use a 7806 instead of a 7805 which provides a good six volt source. Substituting a 7806 for the 7805 will allow the deletion of R2 and R3. The 15 volt one watt zener in combination with R1 provides a degree of protection against noise and spikes on the supply line. It is imperative that you place capacitors between pins 1 and 2 as well as between 2 and 3 regardless of design to prevent oscillation in the regulator.

There are many times the circuit as shown will work properly, but a .1 wF cap placed on each side of the regulator will insure proper operation. This is detailed in the Motorola data books as well as National data books. Note that these capacitors need to be as close a possible to the regulator for proper functioning.

The supply should be able to supply about 400mA of current at six volts with no trouble as the regulator is a one amp device. I would recommend using a one amp fuse on the input for safety and protection.

Larry G. Newson Sr.
Webster, MA

The author of that piece, when we asked him about your concerns, stated that he felt the capacitors were not needed in this case. He said that if this unit were the only regulator in the entire power circuit, then the capacitors would be absolutely required. However, because the Tandy computer already has its own built-in voltage regulator, any power variations would be well within its tolerances (the wall power supply is none to stable in its own right, as anyone who has dealt directly with the electric company will be able to tell you). Hence he left them out to simplify the construction and make it easier to find parts.

Eds

I was the fifth of my family to get a Model 100.

PORTABLE PHONE CRASHES DOWNLOAD

I bought my Model 100 to access CompuServe's aviation weather (GO EMI). I was the fifth of my family and cousins to get a Model 100. My sister got a Model 200 for Christmas. My computer understanding was minimal. I had a problem that caused me to send my laptop back to Radio Shack to have them look at it.

The problem was simple but drove me crazy. I figured that I did not want to download the entire session with CompuServe to conserve memory. When I would press file to download the screen would start printing out undefinable characters followed by a solid block line.

Radio Shack suggested that I check my parameters. I checked my dad's machine and his was set up just like mine. Also mine always worked fine as long as I did not try to download. Finally, I got to snooping around the house and found my portable telephone base station downstairs, where I always used my computer, on telecommunications. I unhooked the phone station from the outlet. Presto, it worked just great.

We love your/our magazine. Dad gave me and my cousin subscriptions for our birthdays. He subscribes also. Keep up the good work.

Everitt B. Dupont
Toughkenamon, PA

Tandy laptop users have more trouble with strange phone hookups than any other group I've seen. It appears that if it isn't a standard simple model telephone hookup when you're experiencing telecommunications problems, then almost always it's the phone system.

Eds

TIC-TAC-TOE CORRECTIONS

It is with great appreciation that I again receive a magazine that supports my Model 100 computer.

The *Tic-Tac-Toe* program in the January, 1988 issue is simple, but fun. I did find a couple of errors in it, though, which had to be corrected before I could make it run. The letter *O* in both line 113 and in line 210 had to be changed to the figure *0*. Line 500 had to be changed to read "PRINT @80". The directions in line 115 had to be changed to read "type 00" to clear the board.

Those changes caused me to wonder if I could improve the program further. Enclosed is a slightly revised and renumbered program which I feel runs a bit better. Double entries are not needed and players are reminded whose move it is.

Warren S. Gibson
Olympia, WA

```
10 CLS:DEFINTA
20 LINE(177,19)-(233,19)
30 LINE(177,35)-(233,35)
40 LINE(197,0)-(197,56)
50 LINE(215,0)-(215,56)
60 PRINT@80," X or O: ";
70 A$=INPUT$(1)
80 IFA$=CHR$(27)THENMENU
90 IFA$="0"THEN10
100 PRINTA$;" at position 1-9:";
110 B$=INPUT$(1)
120 IFB$=CHR$(27)THENMENU
130 IFB$<"0"ORB$>"9"THEN110
140 B=VAL(B$)
150 PRINT@161,"To start new game type 0.":PRINT @241,"Press ESC for Menu."
160 IFB=1THENPRINT@71,A$:GOSUB260:GOTO60
170 IFB=2THENPRINT@74,A$:GOSUB260:GOTO60
180 IFB=3THENPRINT@77,A$:GOSUB260:GOTO60
190 IFB=4THENPRINT@151,A$:GOSUB260:GOTO60
200 IFB=5THENPRINT@154,A$:GOSUB260:GOTO60
210 IFB=6THENPRINT@157,A$:GOSUB260:GOTO60
220 IFB=7THENPRINT@231,A$:GOSUB260:GOTO60
230 IFB=8THENPRINT@234,A$:GOSUB260:GOTO60
240 IFB=9THENPRINT@237,A$:GOSUB260:GOTO60
250 IFB=0THENGOTO10
260 PRINT@80,SPACE$(27)
270 RETURN
```

Listing 1. A revised edition of Tic-Tac-Toe.

For an updated Tic-Tac-Toe, see listing one.

Eds

EMBEDDING PRINTER COMMANDS

I think you may have inadvertently given some incorrect advice in response to Fred Forrester's letter in the I/O section of the Jan., 1988 issue.

Mr. Forrester wanted to know how to embed printer commands in his text files and you printed a short BASIC program to help with that.

Unfortunately, your solution doesn't work. To begin with, Line 20 should read: "20 PRINT#1, CHR$(27)". By leaving you the file number, all the program does is print an escape character to the screen; which has no effect.

Secondly, you can't load a RAM file into a document you are editing with the *F2* (*LOAD*) command. the computer automatically assumes that you want to load from a cassette tape and will hang up until you press *SHIFT BREAK*. Specifying the file as *RAM:ESC* doesn't work either; the load is aborted.

Finally, there is an easy way to put *ESC* sequences in your text documents. Just type: *CTRL P* followed by the *ESC* key. Of course, this method has its own drawbacks (listed on page 60 of the Owner's Manual) but it will work for some types of files.

All in all, a text-formatter of some kind is essential for making the most of your printer with the Model 100.

Michael A. Wilson
California, MD

Don't know what happened to that answer. On looking back at it, it doesn't make sense to us either. Some one must have spaced-out there for a few moments. Sorry for the inconvenience. We'll try our best to prevent this in the future.

Eds

OPTION ROM WARNING

Readers should be aware. WARNING. Traveling Software TSDOS-100 and PCSG Super ROM are not compatible. Problems occur in the spreadsheet. Neither manufacturer has resolved problem

Kevin Marsh
Greenville, SC

OF REPRINTS AND BATTERIES

I have enjoyed your magazine from the first issue!

One of the most valuable articles, was the one giving printer codes to set the printer in the October 13, 1983 issue.

I'm always in a state of panic when I misplace this particular issue. This article would be especially valuable and it might be worth reprinting!

As you know, the 100 does not permit printer codes to be easily imbedded in text. This article utilizes the code and graph keys to send the information to the printer.

Even though we've said it a million times, another won't hurt ...(especially since several readers have been complaining recently.) Most re-chargeable batteries suffer from memory.

Wanting full power, we recharge the batteries often, forgetting that they must be allowed to be fully discharged. Many portable video cameras and the Model 100/102 turn themselves off before batteries are completely discharged.

If batteries are not allowed to completely discharge occasionally, they get use to a top off and will not deliver the full charge potential. A small flashlight works fine for discharging, and the bulb gives a good indicator. A small battery tester is a valuable tool for finding if a particular battery was fully charged, or if it discharged quicker than the rest.

All re-chargeable batteries will fit the 100/102. The small springs have to be slightly adjusted with long nose pliers to fit the dimple in the tip of the battery.

If you need more capacity, Radio Shack sells four D-cell holders very reasonably. Also they sell a plug that fits the battery input on the 100. Observe the polarity (the plus side of the cell holder goes to the outside of the plug.) A flexible wire 2 conductor wire such as a household extension cord works well.

I used broad rubber bands around the holder, but alkaline cells had so much life, I finally just used black plastic tape around the whole holder. When possible, I try to use the electric adapter, but I'll have to admit it's easy to just use battery power.

I suppose the technical articles are fine, but short useful things interest me more!

John Fulton
Indianapolis, IN

If we had a copy of that issue, we might be able to do something about reprinting, but when we took over there were no files on anything previously published. Nor can we find any cache of back issues which might help.

Eds

A FEW MORE CORRECTIONS

I was pleased to see the *PORTABLE 100 Magazine* back in circulation and I have been reading all issues with interest. However, your response to Fred Forrester in the January issue had errors and I felt I should respond quickly.

I have enclosed the letter that I wrote to Fred this week in which I gave him some tips on embedding print codes into Model 100 test files. This letter also points out the errors in your editorial response.

Lorraine Jensen
Santa Barbara, CA

Fred, I read your Letter-To-The-Editor in the January *Portable 100 Magazine*. You have asked a very good question, but unfortunately there were a few errors in their response.

NOTE: In the following paragraphs *CTRL X* indicates the two keys marked *CTRL* and *x*. For example, *ESC* indicates the escape key at the upper left side of the keyboard. *CTRL-P* indicates holding the key marked *CTRL* down while simultaneously pressing the key marked *P*

On page 60 of the big Model 100 manual, there is a note at the bottom regarding using *CTRL P* to embed printer coded in your text. They mention using *CTRL P CTRL O>* to get a decimal 15 embedded in your text. They neglected to mention that you can also use *CTRL P ESC* to embed decimal 27. I have been using this technique for years to put printer commands inside my Model 100 text files while using the TEXT utility. In Table one is a list of some of the embedded commands I use to talk to my Gemini 10X dot matrix printer. Your printer will of course have a different set of commands.

Printer Cmd	Keyboard input	TEXT display	Decimal
Page Eject	<CTRL-P><CTRL-L>	^L	12
Elongated	<CTRL-P><CTRL-N>	^N	14
Condensed	<CTRL-P><CTRL-O>	^O	15
Reset	<CTRL-P><ESC><@>	^[@	27 64
Bold	<CTRL-P><ESC><E>	^[E	27 69

Table 1. Embebbable printer commands for the Gemini 10X printer

Problems arise when you want to print your Model 100 file. You cannot get embedded printer codes to be processed using the *SHIFT-PRINT* option of the Model 100 TEXT program. You must use the *F3* function key and when the *SAVE TO:* message appears you key in *LPT:* to select the line printer as output instead of the cassette output. This is a character-by-character dump to the printer, so there will be no wordwrap capability. As the note on page 60 says, "you will have to format your text and place carriage returns where needed" before you do the printout. So unfortunately, you cannot have *TEXT* wordwrap capability simultaneously with imbedded printer codes. You could write a BASIC program to do this. I did, but the program prints very slowly because it is processing a character at a time.

This is a character-by-character dump to the printer.

Another problem comes up with the null code (decimal zero, binary: 00000000). The Model 100 uses 00000000 to designate special things like end-of-file. The *CTRL P* command will not let you embed a null code in the text. Unfortunately, on some printers, null is used quite often as a printed code. One trick I used was to embed decimal 128 (binary: 10000000, the telephone symbol) in the text via *CTRL P GRPH P*. Then I flip the dip switch on my printer to disable bit 8 (the top bit) and thus code 128 is transformed into null. This is messy, I do not recommend it. Another way is to create character strings using BASIC, put them into the paste buffer and then paste them into the text as needed.

CAUTION;The printer codes zero (binary: 00000000) through nine (binary: 00001001) and the ASCII characters zero "0" (00110000) through nine "9" (00111001) are used differently by the printer. Also, watch out for the letters A through Z versus *CTRL A* through *CTRL Z*. They are very different also.

Back to the Portable 100 editor's response:

1. Line 20 of their program should read: 20 PRINT #1,CHR$(27).
2. Their idea to "Press F2 (load), type ESC, and press enter" does not work because the *TEXT F2* option will try to load from cassette, not RAM. When you follow their directions, things just load up.
3. They did not mention that *SHIFT PRINT* will ignore the embedded printer codes and you must use *F3* and specify *LPT:* to "save" to the line printer while you are inside the *TEXT* program. Also, as I mentioned above, using this technique you lose word-wrap capability.
4. They did not mention the technique of using *CTRL P* to embed printer codes.
5. They forgot to mention the problems with embedding the null code in the text.

I hope that the above ideas give you some help. I have been experimenting with embedded printer commands in Model 100 text for quite some time now. There have also been articles and letters in earlier

Portable 100 Magazine issues about embedded printed codes,. This is where I got some of my ideas.

Lorraine Jensen
Santa Barbara, CA

Thank you for your correct explanation on embedding printer control codes.
Eds

NEW CLOCK

Enclosed is the listing for *CLOCK.BA* which was in *PORTABLE 100* some time ago, showing some modifications which I have made (lines 10-26 and 310-393).

On January 1st, the message *HAPPY NEW YEAR* appears, and various other messages and reminders for birthdays. Line 242 puts my initials on the clock face (but after a couple hours, the hands wipe off the initials). The first thing that appears on the screen when the program is started is a prompt, asking you to enter a short message, or *ENTER*. If you do enter a message, it will appear along with the clock, but if you press *ENTER* (for no message), then a built-in message will appear—* *HAVE A NICE DAY* *. I would like someone to advise me how I could modify my program (after line 26) so that if I wait longer than five seconds to begin *inputting* a message, then the program will goto line 164 as if I had pressed *ENTER* (for no message).

I have several other programs that I can interrupt by pressing *M*. The screen will clear, and then the *CLOCK.BA* program will start up. So I would like for the clock to start in five seconds even if I fail to *input* a message. I know how to do the *FOR T=1 TO 1650:NEXT* but what kind of *IF...THEN* do I use? I think I saw a program like this once, where you were given five seconds to enter a password, or an alarm would sound. But I can't believe I lost the program.

You sure have a great magazine. I don't know what I'd do without it. I bought my Portable 100 in Okinawa in 1983, and had no prior experience with any computer, and have not had any training since, but have picked it up by self-study.

Gene Bordenkircher
Triangle, VA

A timer delay is fairly simple:

```
1010 ZA=0
1020 A$=INKEY$:IF A$ <> "" THEN INPUT "What is your message"; A$: GOTO 1040
1030 IF ZA < 1651 THEN ZA = ZA+1: GOTO 1020
1040 REM rest of program
```

This little routine will check the keyboard 1,652 times to see if you have touched any key. If you haven't touched the keyboard before it finishes, it will drop automatically to the next routine. If you do touch any key, it will ask for your message, put it in A$, exit the counting routine, then continue with the rest of the program.

Eds

```
10 PRINT:PRINT"            ENTER MSG FITTING BOX
15 PRINT"                ( OR <ENTER> )"
20 PRINT"          ┌──────────────────────────────┐"
21 PRINT"          │                              │"
22 PRINT"          └──────────────────────────────┘"
25 LINEINPUT"          :";M$
26 IF LEN(M$)>28THEN10
163 CLS
164 IF M$=""THEN M$=" *  HAVE A NICE DAY  *"
165 PRINTTAB(105-(LEN(M$)/2));M$
170 OM=99
180 RD=3.1415926535#/180
240 LINE(0,0)-(62,62),1,B
242 PRINT@84,"gb"
245 PRINT@254,"DATE:":PRINT@271,"TIME:"
250 FOR OH=0TO11
260     PSET(31.5+29*SIN(OH*30*RD),31.5-29*COS(OH*30*RD))
261     PSET(31.5+30*SIN(OH*30*RD),31.5-29*COS(OH*30*RD))
270 NEXT OH
310 I$=INKEY$
320 IF I$="M"OR I$="m"OR I$=CHR$(27)THEN MENU
330 IF LEFT$(DATE$,5)="12/25"THEN PRINT@171,CHR$(27);"q";"M E R R Y  C H R I S T
 M A S!":GOTO 375
331 IF LEFT$(DATE$,5)="01/01"THEN PRINT@172,CHR$(27);"q";"H A P P Y  N E W  Y
E A R":GOTO 375
332 IF LEFT$(DATE$,5)="01/15"THEN PRINT@175,CHR$(27);"p";" MOM's birthday next "
:PRINTCHR$(27);"q":GOTO 375
333 IF LEFT$(DATE$,5)="02/08"THEN PRINT@171,"JIM, DICK, MAVIS next":GOTO 375
334 IF LEFT$(DATE$,5)="02/18"THEN PRINT@175,"SARAH's birthday next":GOTO 375
335 IF LEFT$(DATE$,5)="03/02"THEN PRINT@175,CHR$(27);"p";" CHARLOTTE's birthday
next ":PRINTCHR$(27);"q":GOTO 375
336 IF LEFT$(DATE$,5)="03/05"THEN PRINT@175,"JENNIFER's birthday next":GOTO 375
337 IF LEFT$(DATE$,5)="04/23"THEN PRINT@175,"NANCY's birthday next":GOTO 375
338 IF LEFT$(DATE$,5)="05/19"THEN PRINT@175,"PAUL's birthday next":GOTO 375
339 IF LEFT$(DATE$,5)="07/08"THEN PRINT@175,"JANET's birthday next":GOTO 375
340 IF LEFT$(DATE$,5)="07/15"THEN PRINT@175,"CHRIS' birthday next":GOTO 375
341 IF LEFT$(DATE$,5)="08/22"THEN PRINT@175,"THERESA's birthday next":GOTO 375
342 IF LEFT$(DATE$,5)="08/23"THEN PRINT@175,CHR$(27);"p";" MARCIA's birthday nex
t ":PRINTCHR$(27);"q":GOTO 375
343 IF LEFT$(DATE$,5)="10/10"THEN PRINT@175,"GARY's birthday next":GOTO 375
344 IF LEFT$(DATE$,5)="10/26"THEN PRINT@175,"BECKY's birthday next":GOTO 375
345 IF LEFT$(DATE$,5)="11/10"THEN PRINT@175,"PAT's birthday next":GOTO 375
346 IF LEFT$(DATE$,5)="11/14"THEN PRINT@175,CHR$(27);"p";" JOHN's birthday next
":PRINTCHR$(27);"q":GOTO 375
347 IF LEFT$(DATE$,5)="12/18"THEN PRINT@175,CHR$(27);"p";" MARY'S birthday next
":PRINTCHR$(27);"q":GOTO375
348 IF LEFT$(DATE$,5)="08/10"THEN PRINT@171,"H A P P Y  B I R T H D A Y !":GOTO
 375
375 PRINT@254,DAY$;", ";DATE$;"  ! ";TIME$
376 LINE(62,40)-(239,40)
377 LINE(176,40)-(176,61)
379 LINE(62,62)-(239,62)
380 IF VAL(MID$(TIME$,4,2))=0M THEN 310
390 IF VAL(MID$(TIME$,4,2))=00 THEN BEEP:BEEP:BEEP:BEEP
391 IF VAL(MID$(TIME$,4,2))=15 THEN BEEP
392 IF VAL(MID$(TIME$,4,2))=30 THEN BEEP:BEEP
393 IF VAL(MID$(TIME$,4,2))=45 THEN BEEP:BEEP:BEEP
420 T$=TIME$
430 LINE(31,31)-(31.5+MX*27,31.5-MY*27),0
440 LINE(31,31)-(31.5+HX*20,31.5-HY*20),0
480 OM=VAL(MID$(T$,4,2))
490 MX=SIN(OM*6*RD)
500 MY=COS(OM*6*RD)
510 OH=VAL(MID$(T$,1,2))
520 HX=SIN((OM/60+OH)*30*RD)
530 HY=COS((OM/60+OH)*30*RD)
570 LINE(31,31)-(31.5+MX*27,31.5-MY*27)
580 LINE(31,31)-(31.5+HX*20,31.5-HY*20)
620 GOTO 380
630 STOP
```

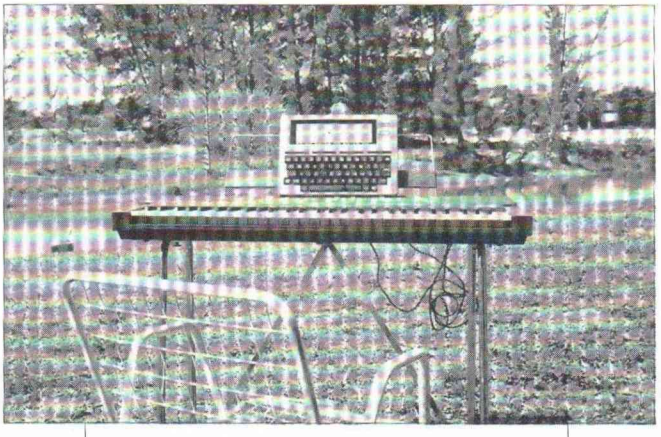
Listing 2. Turn your Tandy 100/102 into a clock with this handy little program.

Duet in Composition: Yamaha and the Tandy 100

Turn your Tandy into a composer's helper.

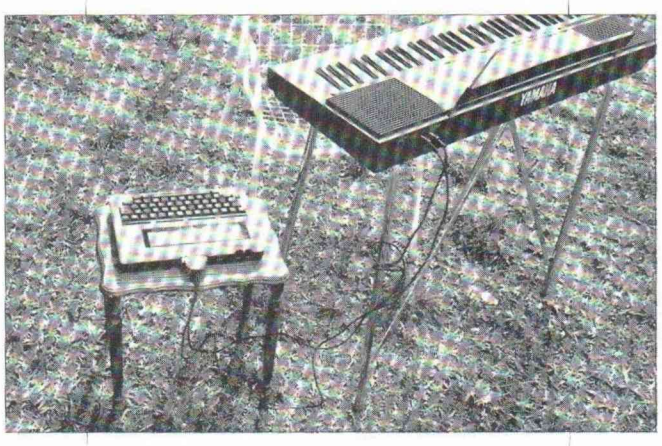
by John Comeau

For the past eight months, I have been attempting to interface my Model 100 to my wife's Yamaha PSR-70 electronic keyboard, in order to develop programs to store, edit, and print the music she composes. Let me quote R. A. Penfold, from his book *MIDI PROJECTS*, to illustrate the problems one encounters in this endeavor:



"The MIDI system is essentially the same as the RS-232 serial ports fitted to many home computers, but in points of detail there are problems which would make it highly unlikely that a serial output from a computer could be used to drive a MIDI input. The first problem is that MIDI interfaces use standard 5 volt TTL levels, and normally have opto-isolator inputs to avoid problems with earth loops. RS-232C ports use signal levels of nominally plus and minus 12 volts (plus and minus 5 volts in the case of the RS-423) with no form of isolation at the inputs. This incompatibility could be overcome with suitable signal processing stages, but there is a more difficult problem with the baud rate. 31250 baud is not a standard rate, and it is not one that can be achieved with any home computer I have encountered. It might be possible to modify the serial interface to operate at the correct baud rate, but this would almost certainly render it unusable for other purposes, and would probably not be a very good idea."

These programs use only six bytes to store each note.



As for the voltage difference, this problem does not apply to the Model 100/102, since the RS-232 signal voltage is plus and minus 5 volts. The negative pulses are removed from the signal by diode D1 (illustration one). And the baud rate? As you will see, this problem is solved with 3 bytes OUTed to the appropriate ports: a simple solution which took months of research to discover. The following two programs are only for the most basic MIDI I/O, and only made to work with a Yamaha PSR-70 electronic keyboard. I am sure, however, that they can be modified to work with any MIDI system.

WHY BOTHER?

The PSR-70, as well as many other modern electronic keyboards, already has RAM storage,as well as save/load capability. Why store the MIDI files on the Model 100? Well, for one thing, these programs use only six bytes to

store each note. With about 28K available in the computer, that's a lot of music; much more than the piano can hold. Also, with modifications to the data and/or programs, the computer files can be played on other electronic instruments (the tapes made directly from the PSR-70 are not MIDI format), or be used to control volume, tempo, and voice besides just the pitch. And of course, one can add many routines to these skeleton programs to print and edit the music. I plan to do this myself as a future project. The most difficult part is done; all of us can use this interface to develop programs of our own.

THE RIGHT BAUD RATE

We'll start with MIDIN.BA (listing one). The MIDI bit stream consists of one start bit, 8 data bits, and one stop bit. No parity bits are generated. The baud rate chosen in line 10 is incorrect, but this will be taken care of in the next line: Port 180 is the 8155 timer register's lower byte. OUTing a 5 into there causes the clock speed of 2.4576 MHz to be divided by 5, yielding 491.52 kHz (the VART chip then divides this by 16 to give a baud rate of 30720, which is within 2 percent of 31250). Port 181 is the upper byte of the timer register. Setting bit 7 with the 64 (40H) puts the output in the *continuous square wave* mode. Port 176 is the command register of the 8155. Bits 6 and 7 or the 195 (OC3H) are the *START* command, without which the baud would remain at 19200, and bits 1 and 2 set ports A and B in the output mode (these ports, which control keyboard, LCD, and other I/O functions, must always be in the output mode). For further info on this chip see Intel's *EMBEDDED CONTROLLER HANDBOOK*, pages 11-31 to 11-44.

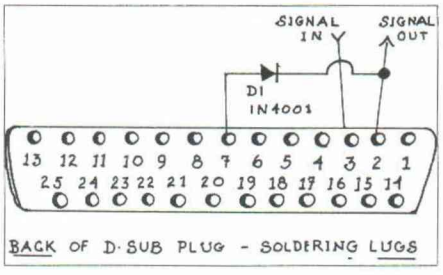


Illustration 1. The Model 100/102 RS-232C -to-MIDI port connections

Lines 40-50: the "N" counter is my cheap way of timing the notes. It works fairly well.

Line 70: Here we split the timer number into two bytes; I could only use 7 bits of each because storing and retrieving chr$(0)'s in text files is a problem.

I'll now give you some help in deciphering MIDI data. Line 100: The first byte of a MIDI data string representing a note is 144 plus the MIDI channel number. Since there are 16, numbered 0 to 15, the largest byte representing a note will be chr$(159). The reason I had to include this filter was a 5 Hz stream of 254's and some initial 255's. I may have to modify this filter in order to receive other types of MIDI commands. The next two bytes are the note value and the velocity byte, respectively. The PSR70 recognizes 60 note values, from 36 to 96. Now, about the Velocity byte. Mr. Penfold's book explains that notes are gated *on* by a 144+channel#, as explained above, and gated *off* by 127+channel#. The velocity byte, then, indicates the level of force with which the keys are struck. However, the PSR-70 ALWAYS has the 144+channel# as the primary byte, and the velocity byte is used to gate the notes on (64) and off (0). Note that I use bit 7 of the note value byte to indicate this status.

RETRIEVING MUSIC

The output program (listing two) will send the music received by the

```
1 CLS:CLEAR1000:DIMN(1000):DIMN%(1000):I
=0
10 OPEN"com:98n1d"FORINPUTAS1
20 OUT180,5:OUT181,64:OUT176,195
30 ONCOMGOSUB100
40 COMON:N=0:PRINT@40,"Start playing..."
:PRINT"hit <SPACEBAR> to exit"
50 N=N+1:PRINT@120,N:IFINKEY$=""GOTO50
60 CLOSE:COMOFF:OPEN"MUSIC.DO"FOROUTPUTA
S1
70 FORJ=0TOI-1:PRINT#1,CHR$((N(J)MOD128)
OR128);CHR$(INT(N(J)/128)OR128);CHR$(N%(
J));:NEXT
80 CLOSE:MENU
100 N$=INPUT$(1,1):IFN$<CHR$(160)THENN$=
INPUT$(1,1):N%(I)=ASC(N$)+2*ASC(INPUT$(1
,1)):PRINT@160,N%(I);:N(I)=N:N=0:I=I+1
110 RETURN
```

Listing 1. The MIDIN.BA program for storing incoming signals on the Tandy 100/102

```
1 CLS:CLEAR1000:DIMN(1000),N%(1000):I=0
10 OPEN"music.do"FORINPUTAS1
20 IFEOF(1)THENCLOSE:GOTO100
40 N(I)=ASC(INPUT$(1,1))AND127:N(I)=N(I)
+128*(ASC(INPUT$(1,1))AND127):N%(I)=ASC(
INPUT$(1,1)):I=I+1:GOTO20
100 OPEN"COM:98N1D"FOROUTPUTAS1:N$=CHR$(
144):OUT180,5:OUT181,64:OUT176,195
110 FORJ=0TOI-1:PRINT@40,N(J):FORL=0TON(
J):PRINT@80,L:NEXT:PRINT#1,N$;CHR$(N%(J)
AND127);CHR$((N%(J)AND128)/2);
120 NEXT:CLOSE:MENU
```

Listing 2. The MIDOUT.BA program for decipering and sending the stored music back to the music keyboard.

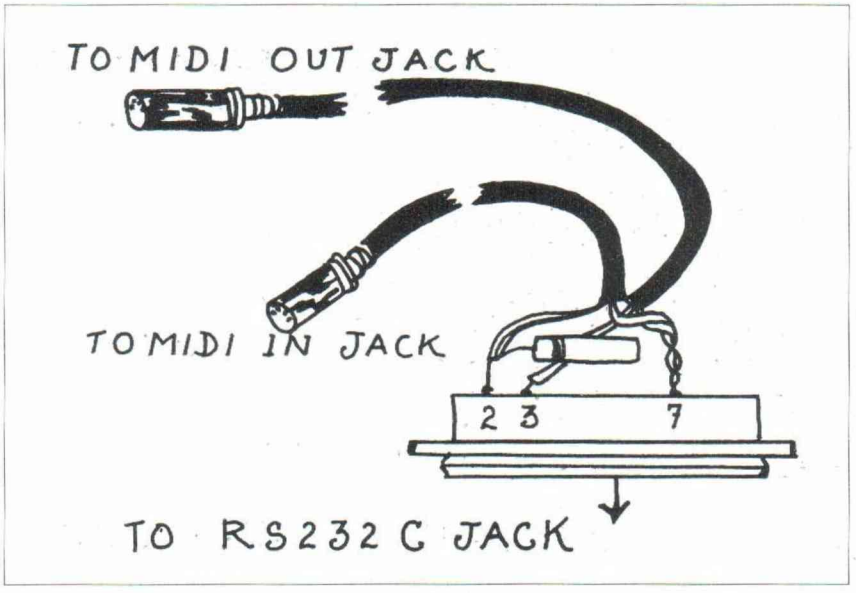


Illustration 2A. The connections between the various hardware pieces.

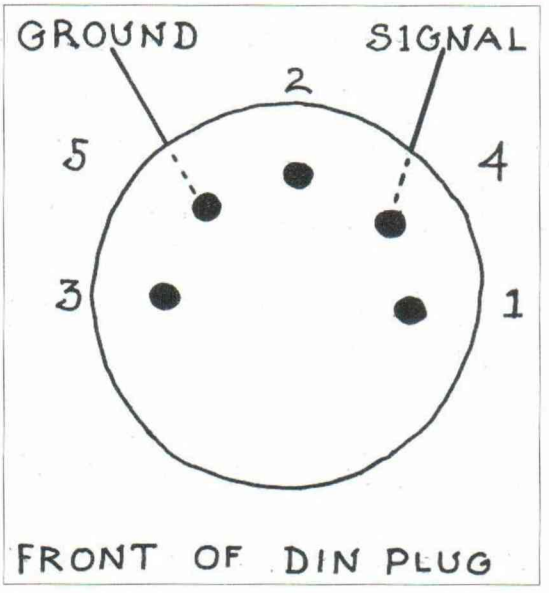


Illustration 2B. Front of the Yamaha MIDI IN/OUT plugs.

program above right back to the piano, at the same tempo.

In line 40 we retrieve the timer value from the two ASCII bytes, using the *AND* operator to get rid of bit 7 which we set in the input program. Likewise, in line 110 we separate the velocity byte from the note value using similar logic. If you wish to use a different channel number just replace the 144 in line 100 with the value you choose.

In order to make the connection this is the procedure. First you need hardware. You will need a male 25-pin D-subminiature connector, two 5-pin male Din plugs, about 12 feet of small-gauge coaxial cable, and one small rectifier diode. Cut the cable in half. Solder an end of one of the cables, and the cathode (banded end) of the diode to pin 2 of the 25-pin connector. Solder the other cable to pin 3. Twist the grounding shields together and solder them both, along with the anode end of the diode, to pin 7. Now, the core of each cable goes to pin 4 of the DIN plugs, and the ground shield is soldered to pin 5. Look closely at the numbers on the plugs; they are not in order. Label the DIN plugs whichever way seems best to you, keeping the following in mind: The cable going to pin 2 of the RS-232 plug outputs the MIDI information from the computer, and will therefore be connected to the MIDI IN jack of the keyboard. The cable going to pin 3 sends data from the piano's MIDI OUT jack to the RS-232 jack on the Model 100-102.

Well, that's the whole story so far. In about three more months I hope to have a music-writing program good enough to be published...but I would rather that someone else beat me to it Good luck.

❑

John Comeau
2011 NW 55 Ave #109-S
Lauderhill, FL.33313
(305) 735-1565
CompuServe 70641,57

REFERENCES:

MIDI PROJECTS, by R.A. Penfold, 1986 available from Electronic Technology Today Inc., P.O. Box 240, Massapequa Park, NY 11762-0240 for $6.95+$1.75 shipping = $8.70. Order #BP182.

EMBEDDED CONTROLLER HANDBOOK, 1987, Intel Literature Sales, P.O. Box 58130, Santa Clara, CA 95052-8130 for $18.00+$1.80 shipping+local sales tax. Order #210918. Or call toll free (800) 548-4725.

RADIO SHACK SERVICE MANUAL: TRS-80 MODEL 100 PORTABLE COMPUTER, CATALOG NUMBERS 26-3801/3802, obtainable (maybe) from Radio Shack. I had to wait three months for mine, and that was a year ago.

ITEM	RADIO SHACK	PRICE
1. 25-pin male D-sub connector	276-1547	1.99
2. (2x) 5-pin DIN plug	274-00 (@1.59)	3.18
3. 12 ft. shielded cable (40-ft roll)	278-751	6.19
4. Switching or rect. diode,	276-1101	.49/pkg of 2

Parts list for the Duet connector.

Make your Model 100, 102, or 200 practically perfect

PCSG has long been the champion of the Model 100 and 200 laptop portable computers. No longer toys, these machines can be made to rival the performance of many desktop models.

PCSG in the past three years has developed software programs and innovations that have rewritten all standards.

We can give you what we regard as the superior Model 100 or 200 system. This is a catalog of what you need to build that nearly perfect portable computer. Full brochures and the answers to any of your questions are just a quick phone call away. So, if you want to add dimensions of portable performance to your laptop unit, call in your order today.

PCSG sells every product on a 30-day full refund trial so there's no risk.

Snap-in ROMS

Super ROM

Get Lucid Spreadsheet, Write ROM, Lucid Database and Thought Outliner all on one ROM. Completely integrated so they all work together with the familiar Cut and Paste working between all applications. Guaranteed to be the finest four programs available for the Model 100, 102 or 200 on one ROM. $199.95, plus shipping.

Write ROM

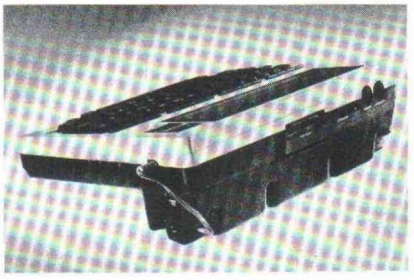
This is full word processing capability like on a desktop. Not only format documents with left and right hand margins, page numbers, headers, footers but phenomenally more. Write ROM gives you memory writing performance: 44 features and functions that really bring text processing power to the Model 100, 102. $99.95, plus shipping.

Lucid ROM

Features like LOTUS 123. But not just a spreadsheet, a program generator as well. Features like Cut and Paste, variable column widths and lightning fast calculation speed. Lucid changes your computer into capacity you never thought possible in the Model 100, 102, $99.95, plus shipping.

Disk + ROM

Now your Model 100, 102, 200 plus any desk top computer is an instant disk system. With a function key transfer any files or BASIC or machine code programs to and from your portable and the other computer. The IBM version works over the phone. $149.95, plus shipping. Cable $40.00.



Business Analyst ROM

Perform those strategic "what if" business problems. Allows detailed financial analyses like Breakeven, IRR, NPV, return on investment, interests costs, PV and others at the touch of a function key. Prints detailed reports. So easy! Also it performs simple everyday calculator functions. Check it out! $99.95, plus shipping. Model 100, 102.

HARDWARE

6-ROM Bank

Access Super ROM, Disk +, Business Analyst and others. The 6-ROM Bank with its 30 hour rechargeable battery pack lets you personally create your ultimate Model 100, 102 system. $199.00, plus shipping.

OTHERS

Custom ROMs

Have your own BASIC program put on to a ROM, or combine your program with a PCSG program. $500.00 minimum order — quantity pricing as low as $15.00 per ROM. Model 100, 102, or 200.

Cassette Programs

PCSG also has cassette programs available at closeout prices. Choose from Tutor+ (typing tutor — $24.98), Type+ (makes your M-100 a memory typewriter — $34.98), Tenky+ (financial calculator — $29.98), Data+ (file manager — $29.98), Sort2+ (sorting routine — $14.98), Write+ (word processor — NEC only — $34.98).

Shipping: $7.50 (U.S.) two days; $15.00 (U.S.) next day.

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines, Blvd., Suite 206, Dallas, TX 75229. 1-214-351-0564.

Circle 78 on Reader Service card.

The Big Picture: SHELL.BA

Make it easier to write custom programs by using this BASIC "shell"

by M. Aiello

When I first saw the Model 102 at Radio Shack I was intrigued, but convinced that anything that small had to be a toy. Real microcomputers ran MS-DOS and were the size of suitcases or better, and what good was a machine that could only handle forty characters of text per line anyway? I resisted for quite some time, but finally the coincidence of a graduate school term paper, and a spring vacation at a beach resort gave me the excuse I needed, and I took the plunge.

Since that time, about a year ago, I have been amazed at the versatility, and general usefulness of the 102. It has gradually become my favorite computer, and for my personal needs and business records it gets used in favor over a Compaq Portable III. It has grown a few appendages as well, such as rom software, a disk drive, a memory expansion device, and at least a dozen critical cables. It serves me as a word processor, secretary, information manager, and portable terminal. What began as an infatuation has become a serious commitment.

The first expansion of my 102's capabilities was the investment in one of the popular multi-program roms that contain a word processing program, an outliner, and some database capability. This acquisition was a major step forward over the native capabilities of the machine, and happened early on.

It is only by programming that your special applications can be accommodated on the 102. Combined with the portability of the machine, and the excellent control of the machine's capabilities *BASIC* gives, a custom application is a powerful tool, well suited to use in the field. All of this makes the effort of programming well worth it.

The solution took the form of a template program

There were a few principal difficulties I encountered in programming in *BASIC* on the 102. The most significant was the difficulty in visualizing the overall organization of a program viewing only a small segment at a time on the small screen. Another problem was dealing with line numbers in a *BASIC* that does not contain a built in renumbering feature, and where stand-alone renumbering programs require cumbersome translations from *.BA* to *.DO* files, and take forever to run. Finally, as I wrote more application programs, I found myself re-writing simple functions, like screens and input routines over and over, with very little variation from program to program.

The solution to these problems took the form of a template program in *BASIC*, which in itself did nothing at all! It did provide a framework upon which any application program could be hung, and eliminated a lot of repetitious coding. Consisting of simple subroutines that performed the housekeeping for any application program, all located at standardized line number locations, the template program allowed me to concentrate my efforts on the application itself.

By allocating blocks of line numbers to various tasks within the program, the template made managing line numbers a little easier. Each service the template provides to the application is always invoked by a *GOSUB* to the same particular line number; this simplifies coding a great deal. The template program provides a standard organization for application programs, and a consistent user interface as well. By using

```
Ø 'SHELL.BA 1Ø/23/87
1 'Copyright 1987, Mike Aiello
2 'General purpose application shell
1Ø '-----------declarations-----------
15 DEFINT A-Z
2Ø DIM PL,II,KX,KP,KV,KN(8),KS(8),KA(8),
EX,EV#,ED
25 DIM BK$,RV$,NR$,FN$,KY$(8),SS$,EV$
5Ø '-----------initializations---------
55 PL=17Ø:BK$="
                ":RV$=CHR$(27)+"p":NR$=CH
R$(27)+"q":FN$="-none-"
1ØØ 'program initialization
1Ø5 GOSUB 29ØØØ 'disp init scrn
11Ø CLS:GOSUB 9ØØØ 'disp stat scrn
15Ø 'init fn keys
155 SCREEN Ø,Ø
16Ø CALL 23161 'clr fn keys
165 KEY OFF
17Ø DATA "    "," "    "," "    "," "
  "," "    "," "    "," "Exit"
175 FOR II=1TO8:READ KY$(II):KX=II:GOSUB
 91ØØ:GOSUB 93ØØ:NEXT II
2ØØ '-------branch to control loop-----
2Ø5 GOTO 32ØØØ
21Ø '---application routines 21Ø-999---
999 '-------main fn key routines-------
1ØØØ KX=1:GOSUB 95ØØ:GOSUB 98ØØ
1Ø2Ø 'KV=Ø:GOSUB 96ØØ ' all keys n/a
1Ø3Ø 'KX=?:GOSUB 91ØØ:GOSUB 93ØØ 'key on
11ØØ '----application code goes here---
11Ø5 'repeat 1ØØØ-1999 as 2ØØØ-2999 for
F2,
111Ø '3ØØØ-3999 for F3 etc. Change KX=2
for
112Ø 'F2, KX=3 for F3 etc.
1899 '----application code ends here---
19ØØ GOSUB 9ØØØ
191Ø 'KV=-1:GOSUB 96ØØ 'all keys av
192Ø KX=1:GOSUB 95ØØ:GOSUB 97ØØ
199Ø EX=-1:RETURN
2ØØØ EX=-1:RETURN
3ØØØ EX=-1:RETURN
4ØØØ EX=-1:RETURN
5ØØØ EX=-1:RETURN
6ØØØ EX=-1:RETURN
7ØØØ EX=-1:RETURN
7999 '--normal exit: unset loop flag---
8ØØØ EX=Ø:RETURN
8998 '--------utility subroutines------
8999 'status display
9ØØØ PRINT@Ø,BK$;BK$;BK$;BK$;BK$;BK$;BK$
;
9Ø9Ø EX=-1:RETURN
9Ø99 'set fn key avail
91ØØ KN(KX)=-1:RETURN
9199 'set fn key not available
92ØØ KN(KX)=Ø:KS(KX)=Ø:KA(KX)=Ø:KEY (KX)
 OFF:KP=28Ø+(KX-1)*5:PRINT@KP,"    ";:RE
TURN
9299 'set available key on
93ØØ IF KN(KX) AND (NOT KA(KX)) THEN KS(
KX)=-1:KA(KX)=Ø:KP=28Ø+(KX-1)*5: PRINT@K
P, KY$(KX);:KEY KX,"":KEY (KX) ON
```

continued

the template approach, all of your application programs will have a similar feel for input and function key use.

The template program that evolved for me is called *SHELL.BA*. It consists of routines that provide an initial program screen, a program status screen, function key support, a file name prompter, and a numeric input routine. It also provides the framework of my favorite program organization.

One of the features I like best about the 102's *BASIC* is the ease of using function key interrupts. Its a simple matter to set up applications as series of function key menus, using a wait loop with the *ON KEY GOSUB* command to jump to selected functions. Since this structure is comfortable for me, *SHELL* is built around it.

SHELL's most interesting component is the set of routines that implement function key menus the way I like to use them. Before getting into the routines themselves, it's helpful to understand the idea behind the way the function key menus work.

In any menu, each function key is either available, or not available. A key that is not available has a blank prompt on the bottom of the screen, and has no effect when pressed. Keys available in the menu are either on, off, or active.

A key that is on is ready to perform some function when pressed; it displays a prompt at the bottom of the screen. When a key is pressed, it becomes active: its prompt is displayed in reverse video, and repeated pressing of the key has no effect until the key's function is complete. When the function is completed, the key returns to the on condition.

Generally, when one key of a menu is active, the other keys must be temporarily disabled; these keys are set off. Their prompts are obscured with pound signs (#'s) until the current function is complete, and the menu is reset. It is possible to construct menus where some keys remain on while a key is active, as the key routines are constructed to allow this.

Taking a look at *SHELL*, you'll notice the *DIM* statements at lines 20 and 25 include all the variables in the program, even those without subscripts. Declaring all variables at the beginning speeds up program execution. Variables used by application code would be added in DIM statements here. The initialization section sets values used later: a forty character blank string, escape sequences to enter and leave reverse video mode, and the default file name string, *-none-*.

Programs based on *SHELL* begin with a *GOSUB* at line 105 to the initial display screen, a subroutine located at lines 29000-29099. This routine performs a screen clear, then puts up whatever introductory message is desired, followed by a wait loop that keeps the screen visible for just a moment.

After the introductory screen, the program status screen is displayed for the first time by a *GOSUB* at line 110. The status screen routine is located at lines 9000-9099.

This screen can contain any information about the application; it is redisplayed at least once with each execution of a function key and is the main source of information to the application program's user.

The next group of lines, 150-175, establish control of the function key menus. The *SCREEN 0,0* turns off the display of the label line, so *SHELL* can control it directly. The *CALL* to the ROM program at address 23161 clears the default settings of the function keys. The *DATA* statement at line 170 contains eight 4-byte strings, which are the key prompts for the main function key menu. Notice that the eighth prompt is *EXIT*; function key eight is reserved in *SHELL* as the normal exit to the system menu.

The *FOR-NEXT* loop at line 175 reads the key prompts into the key prompt array, *KY$()*, makes each key available with a GOSUB to the subroutine at 9100-9199, and turns each key on with a *GOSUB* to the routine at 9300-9399. When this loop is complete, menu handling is set up, and the application can begin.

The *GOTO 32000* at line 205 transfers control to the wait loop for the main function key menu. This loop is placed at the bottom of the program for reasons of speed: placing subroutines nearer to the beginning of the programs causes them to execute faster. The main menu loop flag, the variable *EX*, is set to -1 (true) at line 32000. As long as it remains true, the program will loop on line 32200, waiting for a function key to be pressed. The mechanism used to take a normal exit from the program is simple: pressing function key eight sets *EX* to 0 (false), and the program falls out of the main menu loop, and into the normal exit sequence that follows at lines 32255-32300.

The lines from 1000-1999 represent the standard main menu function key block. One of these blocks must be replicated for each function key that is used in the main menu. The block gets renumbered for each function key; the code for function key two is located at lines 2000-2999, function key three at 3000-3999, and so on.

Each function key block consists of a prologue, application code, and an epilog. The prologue and epilog portions take care of handling the function key housekeeping, with whatever application code desired sandwiched between them. The prologue portion begins with the setting of the key pointer variable *KX* to the current function key number. In the 1000-1999 block, *KX = 1*. The *GOSUB 9500* makes key one active, and the *GOSUB 9800* turns all other keys off. You'll notice at lines 1020 and 1030 two commented-out statements. These lines are used only if another key is to be on while the current key is active. Uncommenting 1020 makes all other keys unavailable, and uncommenting line 1030 and substituting the key number needed for the question mark makes that other key available and on. Line 1030 could be repeated with other values of KX if more than one additional key was to remain accessible.

Lines 1100-1899 are comments that mark where the

```
9310 RETURN
9399 'set available key off
9400 IF KN(KX) AND KS(KX) AND (NOT KA(KX
)) THEN KS(KX)=0:KP=280+(KX-1)*5: PRINT@
KP, "####";:KEY (KX) OFF
9410 RETURN
9499 'toggle key active
9500 IF NOT KN(KX) THEN RETURN
9520 KP=280+(KX-1)*5
9530 IF NOT KA(KX) THEN KA(KX)=-1:PRINT@
KP, RV$+KY$(KX)+NR$;:KEY (KX) OFF:RETURN
 ELSE KA(KX)=0:PRINT@KP,KY$(KX);:KEY (KX
) ON:RETURN
9599 'set all keys availability
9600 FOR KX=1TO8
9620 IF KV THEN GOSUB 9100 ELSE IF NOT K
A(KX) THEN GOSUB 9200
9630 NEXT KX:RETURN
9699 'set all keys on
9700 FOR KX=1TO8
9710 GOSUB 9300:NEXT KX:RETURN
9799 'set all keys off
9800 FOR KX=1TO8
9810 GOSUB 9400:NEXT KX:RETURN
9999 'filename acquire
10000 SS$=INKEY$:IF SS$<>"" THEN 10000
10005 PRINT@PL,"File? ";:FN$=""
10010 SS$=INKEY$:IF SS$="" THEN 10010
10020 IF SS$=CHR$(8) AND LEN(FN$)>0 THEN
 PRINT CHR$(127);:FN$=LEFT$(FN$,LEN(FN$)
-1):GOTO 10010
10030 IF SS$=CHR$(13) THEN IF LEN(FN$)>0
 THEN RETURN ELSE BEEP:GOTO 10010
10040 IF LEN(FN$) = 6 THEN BEEP:GOTO 100
10
10050 IF (SS$ >= "a" AND SS$ <= "z") THE
N SS$=CHR$(ASC(SS$)-(ASC("a")-ASC("A")))
10060 IF (SS$ >= "A" AND SS$ <= "Z") THE
N FN$=FN$+SS:PRINT SS$;:GOTO 10010
10070 IF (LEN(FN$) > 0 AND (SS$ >= "0" A
ND SS$ <= "9")) THEN FN$=FN$+SS$:PRINT S
S$;:GOTO 10010
10080 BEEP:GOTO 10010
10099 'get numeric input
10100 SS$=INKEY$:IFSS$<>""THEN 10100
10105 PRINT@40,BK$;BK$;BK$;BK$;BK$;:PRIN
T@PL,"Value ====> ";:EV$="":ED=0
10110 SS$=INKEY$:IF SS$="" THEN 10110
10120 IF SS$=CHR$(8) AND LEN(EV$)>0 THEN
 PRINT CHR$(127);:EV$=LEFT$(EV$,LEN(EV$)
-1):GOTO 10110
10130 IF SS$=CHR$(13) THEN 10180
10140 IF (SS$>="0" AND SS$<="9") THEN EV
$=EV$+SS$:PRINT SS$;:GOTO 10110
10150 IF LEN(EV$)=0 AND (SS$="-" OR SS$=
"+") THEN :EV$=EV$+SS$:PRINT SS$;:GOTO 1
0110
10160 IF SS$="." AND NOT ED THEN ED=-1:E
V$=EV$+SS$:PRINT SS$;:GOTO 10110
10170 BEEP:GOTO 10110
10180 EV#=VAL(EV$):RETURN
28999 'initial program screen
29000 CLS:SCREEN 0,0
29010 PRINT@122,"* * SHELL.DO - Program
Template * *"
```

continued

```
29020 PRINT@205," Copyright 1987 M. Aiel
lo"
29030 FORII=1TO800:NEXT II:RETURN
31997 '
31998 '---------main menu loop---------
31999 '
32000 EX = -1
32100 ON KEY GOSUB 1000, 2000, 3000, 400
0, 5000, 6000, 7000, 8000
32200 IF EX THEN 32200 'wait for fn key
32250 '
32251 'normal exit
32252 '
32255 CLS
32260 CALL 23164,0,23366 'reset fn keys
32265 CALL 27795 're-init BASIC fn keys
32270 KEY OFF
32275 CLOSE:MAXFILES=1
32300 MENU
```

End of listing one.

application code would go. The epilog begins at line 1900 with a *GOSUB 9000*, invoking the status screen display. The assumption is that any application code executed would cause a change in the program status. Line 1910 is a commented-out line that is used only if any other keys were turned on in lines 1020-1030; this line makes all keys available again.

The *GOSUB 9500* returns the current key from active status; note that this subroutine toggles active status, and this is the same call as in the prologue. The *GOSUB 9700* turns all keys on again, restoring the main menu to its original condition. The prologue ends with the setting of *EX* to -1, to insure that the main menu loop is not exited. This is done because EX could also be used to control a loop for main menu keys that set up their own sub-menus.

The remainder of *SHELL.BA* consists of the utility subroutines. Lines 9000-9090 are the stub of the status display screen routine. To avoid the necessity of constantly re-painting the bottom line of function key prompts, this routine executes a screen clear by printing seven lines of forty blanks, rather than executing a *CLS*.

The function key support routines occupy the lines from 9100-9899. They make use of three flag arrays: *KN(8)* which is true if a key is available, *KS(8)* which is true if a key is on, and *KA(8)* which is true if a key is active. All of these flags are set to either -1 (true) or 0 (false) values, and are used directly in boolean expressions as the conditions of *IF* statements. I had done some experimentation to determine that this approach executed faster than relational expressions in *IF*'s.

The function key routines manage the label line of the display, turn the key interrupts on and off as needed, and set the key status arrays to reflect the current condition of the menu.

The two remaining routines are *prompters*, routines that perform a particular edited keyboard input. The lines 10000-10099 comprise a file name prompter. The routine

```
0 'TIMER.BA 10/23/87
1 'Copyright 1987, Mike Aiello
2 'Simple timer/alarm clock program
30 DIM CT$,CL,HT$,AL,WT$,LL,TF,AF,CN#,CH
#,CM#,CS#,HV#,HN#,HH#,HM#,HS#,WV#,WN#,WH
#,WM#,WS#
60 PP=45:CT$=TIME$:CL=PP+18:HT$=FN$:AL=P
P+58:WT$=FN$:LL=PP+98:TF=-1:AF=0
65 POWER CONT
170 DATA "Set ","Hour","Strt","Clr ","Ex
it"
175 FOR II=4TO8:READ KY$(II):KX=II:GOSUB
 9100:GOSUB 9300:NEXT II
220 IF CT$<>TIME$THENGOSUB260:PRINT@CL,C
T$:RETURN:ELSERETURN
225 IF CT$<>TIME$THENGOSUB260:GOSUB280:P
RINT@CL,CT$:PRINT@CL+80,WT$:RETURN:ELSER
ETURN
230 ON ERROR GOTO 235:GOSUB10100:HV#=EV#
:HH#=INT(HV#):HM#=INT((HV#-HH#)*100):HS#
=INT(((HV#-HH#)*100-HM#)*100):HN#=HS#+(H
H#*60+HM#)*60:GOSUB275:WN#=0:WT$=FN$:RET
URN
235 BEEP:RESUME230
250 ONERROR GOTO255:GOSUB10100:WV#=EV#:W
H#=INT(WV#):WM#=INT((WV#-WH#)*100):WS#=I
NT(((WV#-WH#)*100-WM#)*100):WN#=WS#+(WH#
*60+WM#)*60:GOSUB285:HN#=0:HT$=FN$:RETUR
N
255 BEEP:RESUME250
260 CT$=TIME$:CH#=INT(VAL(LEFT$(CT$,2)))
:CM#=INT(VAL(MID$(CT$,4,2))):CS#=INT(VAL
(RIGHT$(CT$,2))):CN#=CS#+(CH#*60+CM#)*60
:RETURN
270 HN#=CN#+WN#:HH#=INT(HN#/3600):HM#=IN
T((HN#-HH#*3600)/60):HS#=HN#-HH#*3600-HM
#*60:GOSUB275:RETURN
275 HT$=RIGHT$(STR$(HH#),2)+":"+RIGHT$(S
TR$(HM#),2)+":"+RIGHT$(STR$(HS#),2):RETU
RN
280 WN#=HN#-CN#:WH#=INT(WN#/3600):WM#=IN
T((WN#-WH#*3600)/60):WS#=WN#-WH#*3600-WM
#*60:GOSUB285:RETURN
285 WT$=RIGHT$(STR$(WH#),2)+":"+RIGHT$(S
TR$(WM#),2)+":"+RIGHT$(STR$(WS#),2):RETU
RN
300 AF=0:IFHN#=0THENRETURN
310 GOSUB 220:BEEP:FORII=1TO500:NEXTII:S
S$=INKEY$:IFSS$=""GOTO300ELSERETURN
350 AF=0:HN#=0:WN#=0:WT$=FN$:HT$=FN$:RET
URN
4000 KX=4:GOSUB 9500:GOSUB 9800
4100 IFTFTHENGOSUB230ELSEGOSUB250
4900 GOSUB 9000
4920 KX=4:GOSUB 9500:GOSUB 9700
4990 EX=-1:RETURN
5000 KX=5:GOSUB 9500:GOSUB 9800
5100 TF=NOTTF:IFTFTHENKY$(5)="Hour"ELSEK
Y$(5)="Wait"
5110 GOSUB 350
5900 GOSUB 9000
5920 KX=5:GOSUB 9500:GOSUB 9700
5990 EX=-1:RETURN
6000 KX=6:GOSUB 9500:GOSUB 9800
```

continued

will only accept a correctly formed filename, beeping at any key that is inappropriate in a filename. The input filename is returned in variable *FN$* to the calling routine. Similarly, lines 10100-10199 are a numeric value prompter. The numeric value is returned to the calling routine in *EV#*.

There are three areas in *SHELL.BA* where application code can be inserted. The group of lines from 210-999 is reserved for application subroutines. The position of this group of lines near the beginning of the program makes it ideal for frequently called application routines. Application code is added to each of the key routines, at lines 1000-1999, 2000-2999, etc. Finally, code can be added between lines 10199-31999, if you need the room.

The best way to understand how *SHELL* works is to look at an example: *TIMMRG.BA*. *TIMMRG* is a merge file that contains the required code to turn *SHELL* into an alarm clock program called *TIMER.BA*. *TIMER* functions as either a time of day alarm, or a countdown timer.

TIMMRG contains declarations and initial values of the variables used by the timer in lines 30 and 60. Line 65 sets *POWER CONT* so the 102 doesn't power down timing periods longer than ten minutes. The normal default of ten minutes is set on exit at line 32280. Lines 170 and 175 set up the main function key menu with keys four to eight.

The bulk of the application code making up resides between lines 210 and 999. Subroutines at 220 and 225 update the time displays on screen. The routine at 230 gets the time to sound the alarm, using the numeric prompter at 10100. A similar routine at 250 prompts the user for the hours, minutes, and seconds for the countdown timer. The subroutines at 260, 270, and 280 perform the time calculations and set up the alarm and countdown time strings for display. Line 300 marks the alarm sounding routine, which beeps in a loop until any key is pressed. Finally, at line 350 is a simple clear function, that resets a pending alarm.

There is one main function key routine for each key in the main menu, at lines 4000-4999, 5000-5999, 6000-6999, and 7000-7999. Key four invokes the alarm setting function, allowing the user to enter the time of day, or countdown time desired. Key five toggles *TIMER* between alarm and countdown mode. The prompt for this key is modified at line 5100 to reflect the current mode. The routine for key six, the "Start" key, shows an example of enabling another key while active. In this case, the *Clear* function, key seven, is enabled at lines 6020 and 6030 to allow canceling the timer at any time.

The status display screen is formed from lines 9010-9050, and the introductory screen is printed at lines 29010-29020. One unusual patch included in *TIMMRG* are the lines at 9600, 9700, and 9800, which modify some of the key routine loops to run from 4 to 8, rather than from 1 to 8. This change was made because *TIMER* only uses the upper five function keys.

```
6020 KV=0:GOSUB 9600 ' all keys n/a
6030 KX=7:GOSUB 9100:GOSUB 9300 'key on
6100 AF=-1:IFTFTHENGOSUB280ELSEGOSUB270
6105 IFHN#=0THEN6210
6110 GOSUB 9000
6120 IF NOT AF THEN GOTO 6210
6130 GOSUB 225
6140 IF CN# >= HN# THEN GOSUB 300
6150 GOTO 6120
6210 GOSUB 350
6900 GOSUB 9000
6910 KV=-1:GOSUB 9600 'all keys av
6920 KX=6:GOSUB 9500:GOSUB 9700
6990 EX=-1:RETURN
7000 KX=7:GOSUB 9500:GOSUB 9800
7100 GOSUB 350
7900 GOSUB 9000
7920 KX=7:GOSUB 9500:GOSUB 9700
7990 EX=-1:RETURN
9010 PRINT@PP,   "Current time ===> ";:P
RINT@CL,CT$;
9020 PRINT@PP+40," Alarm time ===> ";:P
RINT@AL,HT$;
9030 PRINT@PP+80,"   Time left ===> ";:P
RINT@LL,WT$;
9040 PRINT@PP+160,"   Enter times as HH.
MMSS";
9050 PRINT@PP+200," Press any key to sto
p alarm";
9600 FOR KX=4TO8
9700 FOR KX=4TO8
9800 FOR KX=4TO8
29010 PRINT@122,"* * TIMER - A Simple Al
arm Clock * *"
29020 PRINT@205," Copyright 1987 M. Aiel
lo"
32200 GOSUB 220:IF EX THEN GOTO 32200 'w
ait for fn key
32280 POWER 100
```

End of listing two.

For those readers who are CompuServe users, *SHELL.BA*, *TIMMRG.BA*, and the completed *TIMER.BA* are available through the Model 100 Special Interest Group. Just type *GO M100SIG* at any CompuServe prompt to enter the forum. Within the forum, give the command *DL 4* to get to the Applications Data DL 4 Library. Giving the command *BRO /KEY APPLICATION* will then step you through the downloading menu for the files mentioned in this article. Also available on CompuServe is one other sample program based on *SHELL.BA* called *FINCAL.BA*, not reprinted here because of space limitations. *FINCAL* is a financial calculator based on the HP-12C handheld calculator.

SHELL.BA, has proven to be a very useful programming tool for me, allowing me to quickly assemble application programs that have a polished appearance to them. Of course, *SHELL* represents my programming preferences and style. You can readily adapt the technique to your own needs, by surveying your programs, and extracting your best techniques to make up your own *BASIC* template. ❑

All The Power You'll Ever Need Is Sitting Right In Your Lap.

Traveling Software. A Powerful

BOOSTER PAK

Add 2 Megabytes of Memory to Your Tandy 100/102.

Just snap the BOOSTER PAK onto the bottom of your Tandy 100/102 and plug in two cables. That's all it takes. And you can add up to 2 megabytes of RAM or ROM or a combination of both to your Tandy 100/102. And since BOOSTER PAK is an open system, you can customize it as you go. Add more RAM or ROM software as you need it. BOOSTER PAK comes with: 136K of RAM memory, 64K of ROM software, 16 socketed slots for 32K RAM and/or ROM chips and X-TEL Communication Software. It also has provisions for a battery pak, built-in modem and TS-DOS disk software built-in. It even includes a fast action Asteroids game!

BOOSTER PAK **$429.00**

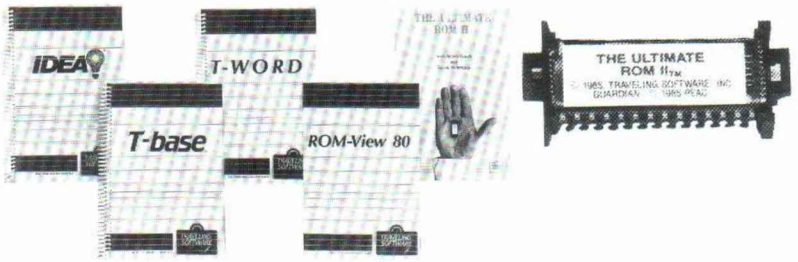
LIMITED TIME SPECIAL: Save $30.00 on IBM or Macintosh BOOSTER LINK software with BOOSTER PAK purchase.

Five Programs On One Chip.

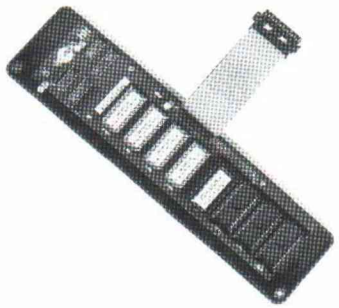
Five applications for the Tandy 100/102/200, and the NEC PC-8201/8300 that occupy only one file on your computer. Programs included are: T-Word—a laptop Word Processor with disk drive support; ROM-View 80—an 80-column display that lets you view 60 characters and includes scrolling; T-Base—lets you create true Relational Databases; IDEA!—a flexible organizer with an outline format; and Portable Disk Drive Support—so you don't have to store TS-DOS in your computer.

THE ULTIMATE ROM II . . . . **WAS ~~$229.95~~**
NOW ONLY $199.00*

NOTE: Tandy 200 version includes TS-DOS instead of ROM-View 80.



THE ULTIMATE ROM II



8-ROM EXPANSION PAK

Add Up to 256K of ROM Software to Your Laptop.

The 8-ROM EXPANSION PAK gives you 8 ROM sockets to plug-in your favorite ROM software. This slim-designed ROM pack measures only 1" thick and 2½" wide. Attaches directly under the Tandy 100/102/200 or NEC PC-8201/8300. Only one connection is required to the existing ROM socket. Buy the 8-ROM EXPANSION PAK with either a four-chip set of SARDINE or Traveling Software's TS-DOS.

What is really amazing is that switching from one ROM software chip to another is done all by software—absolutely no mechanical switches!

BONUS! With each 8-ROM EXPANSION PAK w/SARDINE PLUS purchase, get TS-DOS in a ROM FREE.

8-ROM EXPANSION PAK w/SARDINE PLUS **$299.00**
8-ROM EXPANSION PAK w/TS-DOS in a ROM chip **$169.95**

NOTE: If you already own a SARDINE ROM, you can upgrade it for an 8-ROM PAK for just $129.00. The SARDINE disk version can be upgraded for just $198.00. Call (800) 343-8080 for details.

***Special! Order Now (limited time offer).**

Connection For Your Tandy 100.

SARDINE

Your Laptop Dictionary & Spelling Checker.

Sardine is able to look up a single word in seconds. Or check a 25K document in less than three minutes. It also lets you create your own auxiliary dictionary of terms, words, names and abbreviations. Compatible with the Tandy 100/102/200 and NEC PC-8201/8300 laptop computers. Requires 16K RAM and a Tandy/Purple Computing disk drive. ROM version also includes T-Word Word Processing and text formatting.

SARDINE Disk $99.95
SARDINE ROM ..... WAS ~~$100.95~~
NOW ONLY $119.95*

TS-DOS/TS-RANDOM

TS-DOS—The Traveling Software Disk Operating System

TS-DOS is the complete operating system for Tandy and Purple Computing portable 3½" disk drives. With TS-DOS you can list files both on disk and in RAM, save to disks, load into RAM, kill and rename files. Simply by pressing a function key. With its unique "tag" feature you can load, save or kill several files at once. BASIC programming and disk access included.

TS-RANDOM—Your Rapid Access to Many Laptop Files.

For the Tandy or Purple Computing 100K disk drives. TS-RANDOM retrieves information faster than sequential access methods. Supports up to eight files for simultaneous input and output. Even includes special utilities for quick copying files and recovering erased or damaged disk files. Basic programming and disk access are also included.

TS-DOS Disk (all drives) $69.95
TS-DOS ROM (all drives) $99.95
TS-RANDOM Disk (TDD #1 only) .. $89.95
TS-RANDOM ROM ..... WAS ~~$149.95~~
(TDD #1 only) NOW ONLY $119.95*

MAC-DOS II

Connects Your Mac to Your Laptop or Tandy/Purple Computing Portable Disk Drive.

MAC-DOS II gets your Mac and laptop "talking" by the connection of our three-connector cable. It lets you read, edit and print files created with your laptop as easily as regular Macintosh files. Transfers files between your Tandy M100/102/200 or NEC PC-8201/8300 laptop and any version of the Macintosh at speeds up to 19,200 baud instantly. Also includes a Tandy disk drive adapter for optional direct disk access.

MAC-DOS II with cable $129.95

LAP-DOS II

Connects Your IBM Compatible Desktop to a Tandy 100/102/200, NEC PC-8201/8300 or Tandy/Purple Computing Disk Drive.

LAP-DOS II lets your IBM or compatible "talk" directly to your laptop by the connection of our specially designed three-connector cable. Links Tandy and NEC with all IBM or compatibles including PC, XT, AT and the new PS/2 series. Transfers files at speeds up to 19,200 baud instantly. Also includes a Tandy disk drive adapter for optional direct disk access.

LAP-DOS II with cable $129.95

ORDER TOLL FREE U.S.
1-800-343-8080
in Washington State
206-483-8088
Visa, MC, AMX Accepted

POWERCELL

Power Your Disk Drive or Laptop.

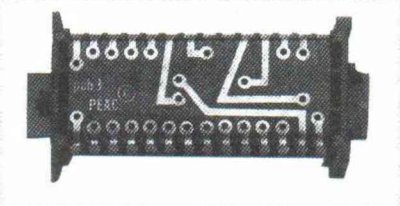
Powercell is the lightweight and convenient way to power your disk drive or laptop simultaneously. A performance far superior to "quick drop" Ni-cad batteries (up to 40 hours of power!). For use with portable disk drives and laptop computers. Including Tandy 100/102/200 and NEC PC-8201/8300.

POWERCELL **$79.95**

CUSTOM SERVICES

We Convert Your Program Into a Plug-in Rom Chip.

Now you can have your own programs burned onto an EPROM so they will work just like the built-in ROM software already in your TANDY 100/102/200 or NEC 8201/8300. It's the ideal way to free-up your RAM memory for data storage and also make your programs copy protected. Our patented Flexible Circuit Board allows you to change your programs and to have your EPROM re-burned at substantially lower prices. Overnight service ensures that we can meet your schedule. Our service is guaranteed or your money back. Quantity pricing as low as $41.00 each. No minimum quantity. Call for your free ROM information kit.



ORDER TOLL FREE U.S. 1-800-343-8080 in Washington State 206-483-8088 Visa, MC, AMX Accepted

BOOSTER LINK

Makes Your Desktop PC Into a Disk Drive For Your Laptop Computer.

BOOSTER LINK allows you to transfer files between your desktop computer and your Tandy 100/102/200 or NEC PC-8201/8300 using a special three-connector cable and TS-DOS or any other disk operating system. It allows TS-DOS to load, save, rename and kill files directly. If you have a BOOSTER PAK or the ROM version of TS-DOS, you can also create and move from subdirectory to subdirectory with the push of a button. BOOSTER LINK comes with a cable and is available for the IBM PC, XT, AT or compatibles and all models of the Apple Macintosh.

BOOSTER LINK MAC/IBM with cable **$99.95**

BONUS! BOOSTER PAK owners save $30.00 (limited time offer)

NOTE: BOOSTER LINK requires disk drive software to operate!

ROM2/CLEASEAU ASSEMBLER

Three Programs on One Chip.

Cleaseau TEXT Editor—increases the Word Processing power of your built-in TEXT program. Append to a paste buffer, search and replace, insert, overwrite, and more; Cleaseau Basic Inspector—shortens programming time by helping you find the error causing statement in a line; ROM2 Macro Assembler and Symbolic 8085 Debugger—assembles 300 lines per minute, allows you to simulate 1000 instructions per second, uses the complete 8085 instruction set, and more.

ROM2/CLEASEAU ASSEMBLER . . **$99.95**

When ordering, please match your computer model with the appropriate order number listed in the following chart. This number will help us expedite your order. Thank you.

Product Name	Tandy 100	Tandy 102	Tandy 200	NEC	Macintosh	IBMs & compatibles
ULTIMATE ROM II	RS1-UR2	RS1-UR2	RS3-UR2	NE1-UR2		
SARDINE Disk	RS1-SD1	RS1-SD1	RS3-SD1	NE1-SD1		
SARDINE ROM Chip	RS1-SD2	RS1-SD2	RS3-SD2	NE1-SD2		
SARDINE 4 ROM Set for Booster Pak	RS1-SD3	RS1-SD3				
MAC-DOS II	AP1-MD1	AP1-MD1	AP1-MD1	AP1-MD1	AP1-MD1	
LAP-DOS II	PC1-LD1	PC1-LD1	PC1-LD1	PC1-LPD1		PC1-LD1
POWERCELL	HW1-PP1	HW1-PP1	HW1-PP1	HW1-PP1		
8-ROM EXPANSION PAK						
w/SARDINE PLUS	RS1-SD4	RS1-SD5	RS3-SD3	NE1-SD3		
w/TS-DOS	RS1-TS3	RS1-TS4	RS3-TS3	NE1-TS3		
TS-DOS Disk	RS1-TS1	RS1-TS1	RS3-TS1	NE1-TS1		
TS-DOS ROM Chip	RS1-TS2	RS1-TS2	RS3-TS2	NE1-TS2		
ROM2/CLEASEAU ASSEMBLER	RS1-CL1	RS1-CL1	RS3-CL1	NE1-CL1		
BOOSTER PAK	HW1-BP1	HW1-BP2				
Options:						
32K CMOS RAM ($20)	HW1-BP3	HW1-BP3				
6 Slot RAM Expansion Board ($69)	HW1-BP4	HW1-BP4				
256K RAM Expansion Modules ($159)	HW1-BP5	HW1-BP5				
Internal Nicad Battery Pak ($69)	HW1-BP6	HW1-BP6				
IBM PC Booster Link w/cable ($69.95)*	PC1-BL1	PC1-BL1				PC1-BL1
Mac Booster Link w/cable ($69.95)*	AP1-BL1	AP1-BL1			AP1-BL1	
Sardine Plus 4 Chip Set ($194.95)	RS1-SD3	RS1-SD3				
BOOSTER LINK (Mac)	AP1-BL1	AP1-BL1	AP1-BL1	AP1-BL1	AP1-BL1	
BOOSTER LINK (IBM)	PC1-BL1	PC1-BL1	PC1-BL1	PC1-BL1	AP1-BL1	PC1-BL1

*BOOSTER PAK owners and purchasers save $30.00. Only $69.95. Limited time offer.

Traveling Software

North Creek Corporate Center • 19310 North Creek Parkway • Bothell, WA 98011

Delightful Data Entry

A simple and easy way to collect data in the field

by Joe Strayhorn

The Tandy Model 100/102 can be a convenient and useful data entry station. I wrote this program to create data files that could be easily uploaded to a desktop computer and analyzed with statistical packages such as SPSS/PC or Systat. The program can be especially useful for researchers. But it is also useful for business or any other situation where data is entered into the computer.

Sometimes the laptop is especially useful as a data entry device because of its mobility: for example, an observer is rating behaviors in a classroom, or is gathering data from archives where a desktop computer is not available. At other times, using the Model 100 as a data entry station is advantageous because it frees up a larger computer for other purposes.

The data entry program printed here is small enough to leave lots of memory left over for data records. And it has a major advantage over some much larger database management programs: it allows checking of each piece of information at the time it is entered. The user can customize it to make sure that each number entered falls within the permissible range for that variable. If an entry does not meet the criteria, the program gives an explanation and a prompt to enter again. I'll soon tell you how to specify the minimum and maximim value for each variable you want to enter. There's also a way of telling the program not to reject some numbers outside the permissible range, so that you can use those numbers as missing value codes.

Laptops are especially useful because of their mobility

In creating files that are easily read by statistical packages, it is desirable that each variable consume exactly the same number of columns for each case. If the amount of information per case exceeds that which fits on one line, the statistical package can read more than one line per case. The statistical package can be told to find each variable in particular columns of a particular line. If the lines for each case are of length less than the 80 columns usually displayed on a desktop monitor, it is easier to read them once they are uploaded. This program starts a new line when the length of a line for one case is about to exceed 78 columns.

Certain variables, such as the subject's name, will naturally vary with respect to how many characters they contain. For such variables it doesn't make sense to insist that the data entry person enter an exact number of characters. Also, there are certain numeric variables where the number of digits can vary—for example, standard scores on tests can consist of two or three digits, depending on how well the person scored. In order to have variables such as these take up the same number of columns for each record, the program has a "padding" convention available, where all the allotted spaces for the variable to the left of what is entered are filled with zeros. For example, if the entry person entered 87 for the standard reading score, and the program allowed three columns for that variable, a zero would be added so that the output file would contain 087 in the three columns, The statistical package

```
10 REM "out.do" is the name of the output file
20 OPEN "out.do" FOR APPEND AS #1
30 REM A$(I) is is the data variable itself; AL$(I) is its label;
40 REM AD(I) is the number of characters for that variable;
50 REM PA(I) is the code for how "padding" takes place
60 REM MN(I) is the minimum for a numerical value
70 REM MX(I) is the maximum for a numerical value
80 REM NV is the number of variables you wish to enter
90 REM Make sure NV is the same as the number of data statements
100 NV=6
110 REM M1 and M2 are missing value codes that won't get rejected
by minimum and maximum checker
120 M1=-9
130 M2=9
140 DIM A$(NV),AL$(NV),AD(NV),PA(NV),MN(NV),MX(NV)
150 RESTORE
160 REM next section is data entry routine
170 FOR I=1 TO NV
180 READ AL$(I),AD(I),PA(I),MN(I),MX(I)
190 PRINT "Please enter ";AL$(I)
200 INPUT A$(I)
210 REM next section is the routine for change of an entry
220 IF A$(I)<>"ch" AND A$(I)<>"CH" THEN GOTO 360
230 INPUT "Number of variable to change";V
240 PRINT "Old value of";AL$(V);" was ";A$(V);". New value?"
250 INPUT A$(V)
260 REM next section checks and pads the changed entries
270   IF   A$(V)=""   THEN   PRINT   "Enter   something   for   this
variable":GOTO 240
280 IF VAL(A$(V))=M1 OR VAL(A$(V))=M2 THEN GOTO 300
290   IF   VAL(A$(V))<MN(V)  OR  VAL(A$(V))>MX(V)   THEN   PRINT   "This
should be between ";MN(V);" and ";MX(V):GOTO 240
300 IF LEN(A$(V))>AD(V) THEN PRINT "This should have no more than
"AD(V);" characters.":GOTO 240
310 IF PA(V)=1 THEN GOTO 330
320   IF   LEN(A$(V))<>AD(V)   THEN   PRINT   "This   should   be
a";AD(V);"character entry.":GOTO 240
330 IF LEN(A$(V))<AD(V) THEN A$(V)="0"+A$(V):GOTO 330
340 GOTO 190
350 REM next section checks the original entries
360   IF   A$(I)=""   THEN PRINT "Please enter   something   for   this
variable.":GOTO 190
370 IF VAL(A$(I))=M1 OR VAL(A$(I))=M2 THEN GOTO 390
380   IF   VAL(A$(I))<MN(I)  OR  VAL(A$(I))>MX(I)   THEN   PRINT   "This
should be between ";MN(I);" and ";MX(I):GOTO 190
390 IF LEN(A$(I))>AD(I) THEN PRINT "This should have no more than
";AD(I);" characters.":GOTO 190
400 IF PA(I)=1 THEN GOTO 420
410   IF   LEN(A$(I))<>AD(I)   THEN   PRINT   "This   should   be
a";AD(I);"character entry.":GOTO 190
420 IF LEN(A$(I))<AD(I) THEN A$(I)="0"+A$(I):GOTO 420
430 NEXT I
440 INPUT "check answers? y/n";C$
450 IF LEFT$(C$,1)="y" OR LEFT$(C$,1)="Y" THEN GOSUB 570
460 REM next section writes values to output file
470 FOR I=1 TO NV
480 L=L+LEN(A$(I))
490 IF L>78 THEN PRINT#1, "":L=LEN(A$(I))
500 PRINT#1,A$(I);
510 NEXT I
520 PRINT#1,""
530 INPUT "another case? y/n";Y$
540 IF LEFT$(Y$,1)="y" OR LEFT$(Y$,1)="Y" THEN CLEAR:GOTO 20
550 IF LEFT$(Y$,1)="n" OR LEFT$(Y$,1)="N" THEN CLOSE:END
560 IF LEFT$(Y$,1)<>"y" AND LEFT$(Y$,1)<>"n" AND LEFT$(Y$,1)<>"Y"
AND LEFT$(Y$,1)<>"N" THEN GOTO 530
570 REM next section allows data entry person to check
580 FOR I=1 TO NV
590  PRINT "Current value of ";AL$(I);" is ";A$(I);". Enter p  to
preserve, a to alter."
600 INPUT D$
610 IF D$<>"a" AND D$<>"A" THEN GOTO 720
620 PRINT "Enter new value"
630 INPUT A$(I)
640 REM next section checks and pads the changed entries
650   IF   A$(I)=""   THEN   PRINT   "Enter   something   for   this
variable":GOTO 620
660 IF VAL(A$(I))=M1 OR VAL(A$(I))=M2 THEN GOTO 680
670   IF   VAL(A$(I))<MN(I)  OR  VAL(A$(I))>MX(I)   THEN   PRINT   "This
should be between ";MN(I);" and ";MX(I):GOTO 620
680 IF LEN(A$(I))>AD(I) THEN PRINT "This should have no more than
";AD(I);" characters.":GOTO 620
690 IF PA(I)=1 GOTO 710
700   IF   LEN(A$(I))<>AD(I) THEN PRINT "This should   be   ";AD(I);"
characters long.":GOTO 620
710 IF LEN(A$(I))<AD(I) THEN A$(I)="0"+A$(I):GOTO 710
720 NEXT I
730 RETURN
740   REM   data   statements   define   variable   label,   number   of
characters,   padding   convention,   minimum,   and   maximum,
respectively
750 DATA dent.number,3,1,0,999
760   DATA   subject's   name   (first   name   first;   underlines   in
between),20,1,0,0
770   DATA   subject's   date   of   birth   (monthdayyear   without
separators),6,0,0,999999
780 DATA subject's sex (0=Female 1=Male),1,0,0,1
790 DATA reading standard score,3,1,0,250
800 DATA conduct rating (0 to 7 scale),1,0,0,7
```

Listing 1. A simple Data-entry program for the Tandy laptop computers.

would read it as 87.

On the other hand, for some other variables, such as the birth date that is a six digit string, one does not wish to pad with zeros. Here, the convention should be for the program only to accept those entries that are the exact number of digits specified.

The program is set up to allow a choice between either of these conventions. If padding to the left with zeros is desired, the variable called *PA(I)* is set equal to 1; if a strict requirement of an exact number of digits is required, *PA(I)* is set equal to 0. I'll soon tell you how to specify the padding convention variable.

When you run this program, the first thing you see is a prompt for the first variable. Type in the value, and if you have entered it within the permissible parameters, you get the prompt for the next variable. If you have too many characters or you're outside the permissible range, you get a polite explanation and a request to enter again.

If, before finishing the entry for a case, you discover that you have made an error and want to change a value, you enter the letters "ch" at the prompt for any variable. The program will respond by asking you for the number of the variable you want to change. (The first variable is number 1, the second one is number 2, and so forth.) You enter that number, and the program prompts you with the old value and a request for a new one. When you enter the new one, the program checks it and returns to whichever variable you left off with.

The program appends the values of each variable in memory to an output file.

After you've gotten through entering all the values for a case, the program asks you whether you want to check your entries. If you answer "y", for yes, then the program displays, one by one, the value you entered for each variable and asks you to type "p" to preserve that value, or "a" to alter it. If you type "a", the program prompts you for a new value. If you've entered a "p", the program remembers it until you change it, so that you can just hit the enter key on subsequent variables to speed up the checking process, if you don't want to change the value.

After this, the program appends the values of each

variable in memory to an output file called *OUT.DO*, and asks if you want to enter another case. If you answer "y", the process begins again; if you answer "n", the data entry is terminated.

When you start the program over again, new cases are appended to the end of the *OUT.DO* file. You accumulate cases until you want to upload the *OUT.DO* file to a larger computer, and then (once you're sure it was uploaded successfully and you've backed it up) you can kill the out.do file and start another one. You don't have to create an out.do file, the program does it for you.

The program written here is for a hypothetical data set where six variables are entered: the subject's name, date of birth, sex, reading standard score, and conduct rating. The convention of padding to the left with zeros is used for the subject's name and the standard reading score; for the other variables, a set number of digits is required. For this data set the values 9 and -9 are missing value codes: these will not be rejected by the range checker.

Here's how to alter this program so as to customize it for your own data set. The changes occur in the *DATA* statements at the end of the program, the *NV* (number of variables) definition in line 100, and the missing value codes (*M1* and *M2*) in lines 120 and 130. For line 100, you simply figure out how many variables you want to enter for each case, set NV equal to that number.

NV should equal the number of *DATA* statements. For lines 120 and 130, you decide on how you want to code your missing values, and set *M1* and *M2* equal to those missing value codes.

Then you set up the *DATA* statements for your own data set. Each *DATA* statement should have five elements, separated by commas. The first is the variable name; this is the name the program uses to prompt you before each entry. Everything up to the first comma is the variable name. The second element is the number of characters allotted for that variable. The number you specify here, before the second comma, is how many columns that variable will occupy in your output file. The third element in the *DATA* statement is the padding convention. If you want the variable padded to the left with zeros, put a 1 before the third comma; if you'd rather be required to enter the exact number of digits, put a 0. The last two elements are the minimum and maximum, respectively, for the permissible values for that variable. The program will reject any entry outside that range, except for the missing values you supply. If the variable is not a numeric one, the value the program will see is 0; you can make the range checker perfect numerical values by entering 0 for both the minimum and maximum for these variables. For local telephone numbers you can enter zero as a minimum and 9999999 as the maximum, so no values will be rejected.

Would you rather not bother with a minimum and a maximum for each variable?

When you enter these five elements in a *DATA* statement for each variable you want, and set *NV* accordingly, and set *M1* and *M2* to whatever you want, you are all set to use the program to enter your own data set.

Would you rather not bother with a minimum and a maximum for each variable? If so, then change a few lines. First, in line 180 — the *READ* statement — eliminate *MN(NV)*, *MX(NV)*, so that the program does not look for these values in the DATA statements. Eliminate lines 290,380, and 670, the ones that check for whether entries are less than *MN* or greater than *MX*. And in the *DATA* statements, just enter the first three elements, and skip the minimum and maximum.

Variables such as names look a little better if they're padded with spaces rather than zeros. I use zeros here, though, because of the way the text editor for the Model 100, in a spirit of helpfulness, thinks that strings separated by spaces are words and thus avoids breaking them up. The text editor adds soft spaces that make the variables appear not to start in exactly the same column if you inspect the *OUT.DO* file. If you pad with zeros rather than spaces, and if you separate elements such as first and last names with underline characters instead of spaces, then you can take a look periodically at your out.do file with the text editor and see the variables lining up in the same columns case after case. If you'd rather use spaces than zeros for padding, make the alteration in lines 330, 420, and 710.

Once you upload your *OUT.DO* file to the desktop computer, it should be an ASCII file that your statistical package can easily read.

Incidentally, the program can also be used in machines other than the Model 100. If you have an IBM compatible, it works fine with *GWBASIC*. You may want to take advantage of its checking routines, or the fact that it costs nothing, and use it on a desktop machine for data entry.

❑

Model 100 Numbers

Your computer isn't as accurate as you think it is. Read on for an in-depth, non-technical look at just how the Tandy computers compute.

by Louis C. Graue

Most calculations are performed on computers without serious error. Nevertheless, all computers and calculators do occasionally produce highly erroneous answers; and when they do, they display them as neatly and authoritatively as correct ones!

For example, type *PRINT (1-7*(1/7))/(1-3*(1/3))* on a Model 100 computer. The computer prints "2" for an answer while hand arithmetic tells you that the result should be 0/0 and therefore undefined. The Model 100 will tell you that *(1-69 * (1/69)) / (1-68 * (1/68)) = 3* and that *3* (1/3) = .99999999999999*.

If you try the above on other computers you will probably get different answers. My Heathkit H89 tells me that *3*(1/3)* is equal to 1. This does not mean that it is better at arithmetic than the Model 100. It just uses a different system for string numbers. The H89 tells me that *5*(3/5-1/2)-1/2=1.19209E-7* while the Model 100 gives the correct result of *0*. It is not likely that your home computer will give your banking balances exactly the same as your bank computer does since they probably do not store numbers alike.

There is no largest real number and between any two different real numbers there are infinitely many distinct numbers. Every computer has the same problem of trying to represent the infinite dense set of real numbers using a finite discrete set of numbers. For any computer you can cook up arithmetic calculations that will yield incorrect results. To understand why they are incorrect and to have a chance of avoiding some serious errors requires a knowledge of how your computer stores numbers.

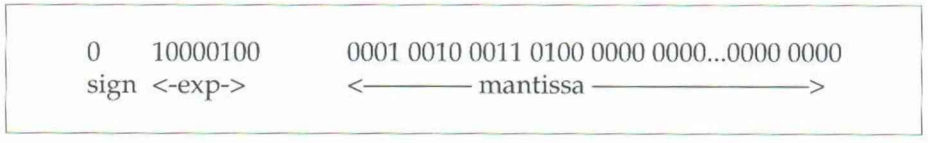


Figure 1. How the Model 100/102 stores numbers in memory.

HOW THE MODEL 100 STORES NUMBERS

The Model 100, using *BASIC* handles variables in a different way than most computers. The default condition is double precision rather than single precision. If you do not specify that a variable is an integer or single precision it is stored as a double precision variable. This computer also stores numbers using binary-coded-decimal (BCD) cells instead of a power of 2 base like must other computers.

We will see that the only numbers *X* stored are those that can be expressed exactly as *X = .d1d2d3d4d6d6d7d8d9d10d11d12d13d14E + c*, where *.d1...d14* is a 14 digit decimal with d1 not zero, and c an integer between -64 and 63. For example: the number 1234 expressed in the above notation would be *1234 = .12340000000000 *10^4*. The number is stored as a 64 bit quantity in memory as shown in figure 1.

The first bit represents the sign of the number; 0 is plus and 1 is minus. The next 7 bits give the exponent c, with a bias of 64. This means that the exponent is obtained by subtracting 64 from the value stored in these bits. In the example 1000100 base 2 is 68 base 10. Subtracting 64 gives the exponent 4. The mantissa is stored in BCD form. That is, each decimal digit is represented by a nibble (four bits). There are 14 of these nibbles: 0001=1, 0010=2, 0011=3, 0100=4, 0000=0, and

so on.

The computer automatically normalizes the mantissa M of a nonzero *X* by *floating* the decimal point immediately to the left of the leftmost nonzero digit of X and then adjusting c accordingly. That is why this is called the *normalized floating point decimal representation* of *X*. As a result of normalization, the smallest mantissa used for a nonzero *X* is *.10000000000000=10^-1* and not *.00000000000001=10^-4*. Now we can figure out the largest and smallest positive numbers and also calculate exactly how many different numbers can be stored in the Model 100. The largest exponent is obtained when the 7 exponent bits are all 1's (1111111) and that is 127 in base 10 arithmetic. Since it has a bias of 64 we must subtract 64 and thus obtain 63 as the largest possible exponent. The largest positive number that can be stored is *L=99999999999999E+63*.

The smallest exponent is given when all the exponent bits are 0's .Subtracting the 64 bias give -64. Therefore the smallest positive number that can be stored is*s=.1E-64*.

For any computer you can cook up arithmetic calculations that will yield incorrect results.

Listing 1 is a program that will let you see exactly how any number you enter is stored. Use it to experiment with large and small numbers with both plus and minus signs.

The\positive storable numbers in Table I. The first number is the smallest and the last is the largest. There are 9*10^13 numbers in each row and 128 rows. In the first row the numbers are 10^-78 apart, in the second row the spacing is 10^-77, and in the last row the spacing is 10^49. Any number not in the list is represented by the number in the list closest to its value. For example 1/3 is between *.333 333 333 333 33E-00* and *.333 333 333 333 34E-00*. The computer selects the first number since it is closer. One-hundred-trillion (100,000,000,000,000) can be stored but one-hundred-trillion-one (100,000,000,000,001) can not be stored. The hundred-trillions that can be stored are 10 units apart. 100,000,000,000,010 and 100,000,000,000,020 and 100,000,000,000,030 and so on can be stored but no values in between. If you write a program where you increment the variable *K* and ask it to stop when *K=100,000,000,000,005* the program will never end because that value can not ever appear.

```
1 'READ NUMBER STORED
10 INPUT"NUMBER TO BE STORED ";N
15 C=PEEK(VARPTR(N))
17 PRINT"CHARACTER =";(CMOD128)-64;" & "
;
18 IF C>=128 THEN PRINT"SIGN IS -" ELSE
PRINT"SIGN IS +"
19 PRINT"MANTISSA IS:"
20 FOR I=2 TO 8
30 A(I)=PEEK(VARPTR(N)+I-1)
40 PRINT INT(A(I)/16);A(I) MOD 16;
50 NEXT I
```

Listing 1. This program reads numbers exactly as they are stored in the computer's memory.

```
.10000000000000E-64, .10000000000001E-64, ... , .99999999999999E-64
.10000000000000E-63, .10000000000001E-63, ... , .99999999999999E-63
                              .
                              .
                              .
.10000000000000E-01, .10000000000001E-01, ... , .99999999999999E-01
.10000000000000E-00, .10000000000001E-00, ... , .99999999999999E-00
.10000000000000E+01, .10000000000001E+01, ... , .99999999999999E+01
                              .
                              .
                              .
.10000000000000E+62, .10000000000001E+62, ... , .99999999999999E+62
.10000000000000E+63, .10000000000001E+63, ... , .99999999999999E+63
```

Table 1. A list of all the positive storable numbers.

SOURCES OF ERRORS

In double precision mode we have to expect roundoff error in the 14th digit. Unfortunately the errors can propagate to the extent that answers can be complete garbage.

Subtracting two nearly equal numbers can destroy accurate significant digits in a single operation! To illustrate, consider the computation of *1-7*(1/7)* on the Model 100. Note that *7*(1/7)=.999 999 999 999 98*
has 14 significant digits with roundoff error in the 14th or least significant digit. This is close to 1. We subtract from 1 and get *1-7*(1/7)=.000 000 000 000 02*.

Notice that the most significant digit is now the inaccurate 2 in the 14th decimal place and the accurate significant digits have been canceled.

Errors can be magnified by multiplying by a large number. The Model 100 tells you that *(10^16)*(1-7*(1/7))=200* instead of the correct answer of *0*.

Another source of error is the adding (or subtracting) of two numbers whose magnitudes are so different that the resulting sum (or difference) gets rounded to the one having the larger magnitude. For example, on the Model 100 *1E14 + 1=1E14*.

That is, one hundred trillion plus one is still one hundred trillion. Also, *1 + 10^-14=1*.

```
1234567*678912=
(1*10^6+234*10^3+567*10^0)*(678*10^3+912*10^0)=
(1*10^6+234*10^3+567*10^0)*678*10^3+(1*10^6+234*10^3+567*10^0)*912*10^0=
(1*678)*10^9+(234*678+1*912)*10^6+(567*678+234*912)*10^3+(567*912)*10^0=
678*10^9+159564*10^6+597834*10^3+517104*10^0=
838,162,351,104
```

Figure 2. The program in listing 2 uses the principles of the Distributive Law of Multiplication to provide accurate, high-precision multiplication that avoids the pitfalls inherent in the computer's built-in multiplication routines.

It is not likely that a programmer can always anticipate when the above sources or errors will be significant. It is always necessary to have some rough idea of the values the computer should be expected to produce in a given application and to check this against the output.

MULTIPRECISION MULTIPLICATION.

A computer is a versatile tool and can do just about anything if you can figure out the right program. Listing 2 is a program that makes it possible to multiply nearly any two numbers and produce an exact answer. The numbers to be multiplies are not stored in the usual way. They are read as string variables A$ and B$. The product is placed in the output string variable C$. The underlying algorithm is based upon repeated applications of the Distributive Law of Multiplication: *A(B + C)=AB +AC* (see figure 2. for examples).

Products are performed in three digit segments and appropriately grouped. If the numbers you wish to multiply contain decimals just enter the digits without any decimal points. After the product is computed then enter the decimal point by hand. The number of decimal places in a product is the sum of the decimal places in each factor. You could also modify the program to automatically place the decimal point.

Here is an example you can use to test the program after you have keyed it in: *123456789012345 * 543210987654321*.

The answer you get should be: *067 063 084 292 027 052 277 861 592 745*.

If you don't have much memory on your machine, you may have to reduce the numbers in the *DIM* statements in line 10 of Listing 2. That will of course reduce the size of the numbers that you can multiply. If you have several other programs stored in your memory you will most likely get an *OM* error when you try to run the program.

❑

```
10 X=0:Q=X:K=X:L=X:S6=X:L1=X:L2=X:U1=X:U
2=X:I=X:J=X:S=X:Y=X:W=X:W1=X:D$="":G$=""
:DIMA$(400),B$(400),C$(400),A(200),B(200
),C(200),D(200):INPUT"MULTIPLICAND => ";
A$:INPUT"MULTIPLIER   => ";B$:GOSUB20:PR
INT"PRODUCT IS  ";C$:END
20 Q=0:C(0)=0:D(0)=0:C(1)=0:FORK=1TO200:
C(K)=0:D(K)=0:NEXTK:L=0:S6=0:GOSUB50:L1=
L2:U1=U2:FORK=1TOU1:A(K)=B(K):NEXTK:D$="
 ":G$=" ":A$=B$:FORK=1TO200:C(K)=0:D(K)=
0:NEXTK:L=0:S6=0:GOSUB50:FORI=1TOU1:FORJ
=1TOU2:C(I+J)=C(I+J)+A(I)*B(J):NEXTJ:NEX
TI:S=U1+U2
30 FORK=2TOS+1:D(K-1)=C(K)-INT(C(K)/1000
000)*1000000-INT((C(K)-INT(C(K)/1000000)
*1000000)/1000)*1000+INT((C(K-1)-INT(C(K
-1)/1000000)*1000000)/1000)+INT(C(K-2)/1
000000):NEXTK:FORK=1TOS:D(K)=D(K)+INT((D
(K-1)-INT(D(K-1)/1000000)*1000000)/1000)
:NEXTK
40 FORK=1TOS:D(K)=D(K)-INT(D(K)/1000000)
*1000000-INT((D(K)-INT(D(K)/1000000)*100
0000)/1000)*1000:NEXTK:FORK=1TOS:GOSUB90
:NEXTK:RETURN
50 L2=LEN(A$):X=INT(L2/3):Y=3*X-L2:IFY=0
THENU2=XELSEU2=X+1
60 FORK=1TOU2:W=L2-3*K+1:IFW<=0THENW=1
70 IFL2-3*(K-1)>=3THENW1=3ELSEW1=L2-3*(K
-1)
80 B(K)=VAL(MID$(A$,W,W1)):NEXTK:RETURN
90 X=D(S+1-K):S6=X+S6:IFS6=0THEN210
100 D$=STR$(X):IFX>99THEN160
110 IFX>9THEN150
120 IFX>0THEN140
130 G$="000":GOTO170
140 G$="00"+MID$(D$,2,1):GOTO170
150 G$="0"+MID$(D$,2,2):GOTO170
160 G$=MID$(D$,2,3)
170 IFQ=0THEN180ELSE190
180 C$=G$:Q=1:GOTO200
190 C$=C$+G$
200 C$=C$+" "
210 RETURN
```

Listing 2. PMULT.BA makes it possible for you to multiply almost any two numbers and get an accurate answer.

The Phantom 24K Loss Of Memory In Your Model 100

by Carl Oppedahl

Dave Ketchum of Santa Monica wrote me recently to tell of a strange and frustrating experience with his 32K Model 100. Every so often he turns on the computer, only to see a brief glimpse of the normal main menu, then suddenly a change to the dreaded January 1, 1900 and a surprisingly small number of free bytes. This sequence of events has happened to me as well, and the cause, a loose or bad memory chip, is usually easy to determine and repair.

MEMORY DESIGN

The microprocessor in your Model 100 (an 8085 CPU) can control up to 64K of memory. The circuit board holding the CPU and the memory chips controls how the memory is arranged. The bottom 32K is in a ROM chip called the *standard* or *operating system* ROM, while the top 32K, if installed, is in RAM.

When the 100 is first manufactured, the RAM chips are filled with random data. This is not a problem because the CPU, when powered up, starts executing instructions located in the ROM. (The ROM contents are determined in the factory, and remain unchanged thereafter despite power being off during shipment.)

RAM chips vary their contents from time to time, and under certain circumstances lose all their contents. Just like someone moving into a new house who puts paper towels in the kitchen and art on the walls. When the 100 is turned on for the first time the CPU copies certain information from the ROM into the previously random RAM. Among the items set up in RAM are the names of the built-in-files, *BASIC*, *TEXT*, and so on. The part of RAM that keeps track of the time of day gets initialized to *January 1, 1900*, a Sunday. The initialized area, located between 62960 and 65535 decimal, is often called the *system* RAM.

The area between 62960 and 65535 is often called the *system RAM*

One of the things the CPU does in that initial power-on sequence is to check to see how much RAM has been installed. In a Model 100, the answer is 8K, 16K, 24K, or 32K. (In a 102, the answer is either 24K or 32K.) The answer to that question is itself stored in RAM (for you hackers, it is at FACO hex). That information is relied upon quite often in the Model 100, such as in displaying the number of free bytes.

In a 100 or 102 the number of free bytes you will see upon such an initial power-up is 29638 for a 32K machine— of the 32768 bytes available, 3130 are used for system purposes (2575 of them in the system RAM area) leaving the 29638 bytes free. If the computer was an 8K model the number of free bytes is far fewer—8192 minus 3130, or 5062 bytes free.

I have described what happens when the computer is powered up for the first time ever. If you are like most of us, you set the date and time, and create a file or two. Then suppose you turn the power off (and by this I mean the ON/OFF switch on the right side of the computer). When you turn the power on, why doesn't the CPU again load initialized values into system RAM from ROM (which would set the date to January 1)?

Putting the question differently, how do you explain the fact that

when you turn the computer on your files are still there and the date is correct?

The answer is that whenever the CPU is powered up, it checks certain things in the system RAM area to figure out whether the computer is just now being turned on for the first time ever (in which case the CPU will initialize all of RAM), or whether the computer has been turned on before (and the existing system RAM contents should be kept).

If I have presented the problem correctly, you should now be asking yourself "how does the CPU, on power-up, decide this very important matter?" The stakes are obviously very high. If the CPU made the wrong decision you would either have a computer that would not run, or would lose all your files.

WRONG TURNS

The CPU could decide wrongly that RAM doesn't need re-initializing when in fact it does, and the result on power-up would be a crazy-looking or a blank screen. But in any event it would be a screen that did not look like a normal main menu. (When this happens, your only choice is the *CTRL-BREAK* reset described below.) Or the CPU could decide to re-initialize system RAM at a time when reinitialization was not called for, showing you a screen full of blank file names where your precious filenames once appeared.

THE DECISION

Here's how the CPU, on power-up (or on your pressing the *RESET* button), decides whether to re-initialize system RAM. It first checks to see whether the *CTRL* and *BREAK* keys were held down at the moment power was turned on. If the answer is *yes*, then off you go to January 1, and a freshly-reloaded system RAM.

Assuming *CTRL-BREAK* has not been pressed, then the CPU checks to see whether the filename *BASIC* is up there in the menu directory in system RAM where the CPU would expect it to be (hackers: it is around F962 hex). If those five letters are missing, or have been changed to lower case, or have otherwise been tampered with in any way, off you go to January 1.

Next the CPU looks to see how much RAM is physically present. By way of background, if there is an empty RAM socket, that part of memory tests out as all 1's in the Model 100, no matter what the CPU puts there. If there is a socket with RAM installed, that part of memory will contain whatever the CPU has put there. The CPU tries reading and writing a few ones and zeros to each socket, to see how many RAM chips are plugged in. The *worst case* is that the computer has only 8K of RAM, so the CPU first checks to see if the chip is plugged in that would bring the total to 16K. (The socket is labelled "option RAM #1".) If it is, the CPU looks for the chip that would bring the total to 24K (option RAM #2), and after that the chip that reaches 32K (option RAM #3).

The CPU could decide wrongly that the RAM doesn't need re-initializing

If the CPU finds that option RAM #1 is missing, it does not bother to look for the next two chips. Similarly if the option RAM #2 is missing the CPU does not look for the last one. The CPU is only interested in contiguous RAM.

Having gotten an answer to the question of how many RAM sockets are filled, the CPU compares the answer with the result that was found the *last* time the computer was turned on (stored at FACO hex). If the two values are the same, the CPU goes on to display the familiar main menu with your filenames and today's date. If, however, the two values are different, then the CPU re-initializes system RAM and shows you an empty menu and a date of January 1.

This explains why you lose all your files whenever you add RAM to your computer.

MR. KETCHUM'S PROBLEM

By now you can probably guess the cause of Mr. Ketchum's problem. It must be that his option RAM #1 chip was loose, or had a bent pin that touched the socket intermittently, or was defective. Thus the CPU, upon power-up, decided the computer contained only 8K rather than 32K. It re-initialized system RAM, and showed only 5062 bytes.

Mr. Ketchum turned the machine off and on a few more times, and got back to the more usual 29638 bytes free. That's because option RAM #1 got working again. When I had this problem on my 100 I carefully removed my option RAM #1, and sure enough one of the pins was bent, and had never extended down into the socket. It did, however, barely touch the socket. This fragile connection took two years to reveal itself. I straightened the pin and gently reinserted the chip.

A chip that is defective internally or a cracked circuit-board trace could do the same thing. In the former case, you can still replace the chip and fix it. In the latter case, which is quite unlikely, you would have to replace the circuit board.

Model 100s rolled off the production line in two varieties, 26-3801's with 8K of RAM and three option RAM sockets, and 26-3802's with 24K soldered in and only one option RAM socket. (Some 3801's had two more chips added so they could be sold as 3802's) Without having seen

it, I can say that Mr. Ketchum's computer is of the former type.

Mr. Ketchum says someone mentioned to him that the problem might be that the built-in nicad (the one that keeps your files intact when you change batteries) "might be getting low." That can't be the explanation, because the nicad can only run low if the AA cells were to run down. But if the AA cells had run down he would not be able to turn on the computer. In any event a problem with run-down AA cells and nicad would result in an empty computer with 29638 free bytes, not 5062.

THE TANDY 102

Can this happen to the Tandy 102? It can, though it is less likely and the solution is less expensive. For one thing, three of the four 8K RAM banks are soldered in place. The only socket is the bank that gets you from 24K to 32K. Thus there are fewer opportunities for the phantom-memory problem to occur. (If it does happen you will see 21446 bytes free instead of 29638.) The one chip you can replace (the one that gets you from 24K to 32K) is a one-piece 64K-bit integrated circuit costing only a couple of dollars. If you happened to get a bad one, the decision to replace it would not be very costly.

The moral of the story is always back up your files

The moral of the story is always back up your files onto tape or disk. There are plenty of ways to lose a file other than those described here, and you should protect yourself.

f you find yourself at January 1, 900, the files may still not be lost. he reinitialization happens to the ystem RAM area, which includes he filenames, but the contents of the iles were in non-system RAM, and ay still be there. BA and CO files are ifficult to recover, but DO files are asy to get back. See *Reviving a owned System*, July 1986 about one ay to recover them onto cassette tape, or use RESTOR, a program distributed free by PCSG that restores them right in RAM.

But make frequent backups just to be safe anyway. ❑

A Dynamic Duo

A review of Peptone's SPRINT.CO and SEARCH.CO

by Alan L. Zeichick

Those expensive high-powered ROM-based programs aren't for everyone. Not every Tandy laptop-computer user needs a spreadsheet, sophisticated word processor or other desktop-style utility in his or her laptop.

If your Model 100, 102 or 200 is nearly perfect right out of the box, without those complicated add-ins, perhaps you should take a look at Pepton's *SPRINT* and *SEARCH* software. They add often-requested features at a reasonable price, and without changing the whole character of the computer.

FINDERS KEEPERS

Although TEXT has most essential text-editing functions, a search-and-replace facility is sadly lacking. Although it's not difficult to write a search-and-replace program in *BASIC*, Peptone hopes you'll choose to purchase theirs instead. And with good reason: *SEARCH.CO* is fast, compact, and offers features difficult to implement in *BASIC*.

Once *SEARCH* is loaded from cassette or disk, the 1,825-byte utility can be run from the main menu, after a *CLEAR 0,60900* is executed from *BASIC*. The program offers a four-option menu: search and replace, numeric code charge, global search with word count, and exit.

The search-and-replace choice will probably be used most often. Once that option is selected, *SEARCH* displays a list of *.DO* files, and requests a file to operate on. Next, the program asks if the search is to be case-sensitive; that is, will a search for *SMITH* accept *smith* or *Smith* as a match? That's a feature difficult to program efficiently in *BASIC*.

> They add often-requested features at a reasonable price.

Finally, the program prompts for the actual search and replacement strings. The strings may be as long as 30 characters. Once they're entered, the search process begins.

Since global search-and-replace is a potentially destructive operation, I would have preferred that *SEARCH* ask one more question, such as "Is that okay?", before actually changing documents, in case the user made an error when specifying the replacement string.

When the search is complete, *SEARCH* returns to its menu. I would have appreciated a summary report of how many changes were performed—if any.

SEARCH's numeric code-change is a more limited utility, changing any single ASCII character within a file into any other single ASCII character, such as changing all occurrences of decimal *090* to decimal *169*. The utility can also delete all of occurrences of *090*, if desired.

The third menu choice, global search and word count, performs the search-and-replace function on *all .DO* files in RAM, without even an "Are you sure?" This is a feature I consider dangerous. The routine does, however, return the number of matches for each *.DO* file, along with each document's word count. Now, why couldn't that have been done for the normal search operation?

Finally, pressing the letter *E* exits *SEARCH*, and returns to the main menu. Pressing *F8* doesn't exit to the menu. This is clearly a non-user friendly feature, since the ROM-based software, as well as many

commercial products, use *F8* for that purpose.

The *SEARCH* program does its job adequately. Throw in the program's uninspired three page documentation, and you have $18.75 worth of software. It's a good value.

PRINTERS PEEKERS

I find the full name of Peptone's *SPRINT* software misleading. It's called a "universal print driver program"; I would call it a simple print formatter.

Hyperbole aside, the 2,708-byte *SPRINT* corrects the obvious shortcomings in the old Tandy 100/102 *TEXT* routines. *SPRINT* offers user-controllable page size, line length, left margin, page numbering and line-end code, as well as in-text formatting commands.

SPRINT does a good job with basic text-formatting functions. Its page-numbering option is welcome, with a choice of page-number formats: at the end of the page, on the right margin; or at the end of the page, centered between the margins and surrounded by dashes(i.e., -3-).

SPRINT's main menu offers five choices: modify setup, soft review, printing a file, printing part of a file, and exit. Again, exit requires that the user press *E*; *F8* isn't implemented.

The various printout options mentioned above are changed by selecting *modify setup* from the menu. The user's preferences may be made permanent by modifying the machine-language program. The user can decide to use either the parallel or serial port for printing. And addressing the dreaded line-feed problem, the user can also choose to terminate printed lines with either a carriage return or a carriage-return/line-feed combination.

Several options may also be temporarily overridden by inserting special codes into the print document; these include line spacing, line length, left margin (which Sprint calls the "indent"), and choice of ragged-right or ragged-left text.

SPRINT offers two additional features. Correction printing allows the user to print a single page, by page number. This is aimed at those with slow printers, who discover a typographical error on page 12 of a 13 page document.

The second "extra" is referred to as soft reviewing. It's what other vendors call *pixel* plotting. A miniature version of a complete page is displayed on the Model 100's screen, with each pixel blackened to represent each character to be printed. Unfortunately, the pixel plot starts towards the center of the screen; there's no way of telling where the text is in relation to the page's absolute left margin. On the whole, the feature is a useful one.

SPRINT comes with a three and a half page laser printed manual. The documentation is insufficient for learning the program. The various page-formatting options aren't adequately explained, and thus the new user will have to experiment to learn what *SPRINT* can do. This could be disappointing for the novice Model 100 owner.

SPRINT is a good program. With its small RAM overhead and reasonable price, it's a good choice of a stripped-down text formatter. ❑

Alan L. Zeichick, formerly the technical editor of Portable 100 *is the technical editor of IDG/Peterborough's* Portable Computer Review *and* CD-ROM Review.

Peptone Software
10112 Ebenshire Court
Oakton, VA 22124

SPRINT.CO version 13.0, $28.95
SEARCH.CO version 13.0, $18.95

Peptone's software is available for Radio Shack Model 100/102 and also for the Model 200.

SEARCH and *SPRINT* were revised in 1987 and Peptone offered its customers the upgrade for only six dollars per program. Because they are very concerned about customer satisfaction, they do this each time they put a new version on the market.

-Eds.

Multi-tasking & Background Printing

Multi-tasking, parallel processing, background printing, print spooling, etc., all have one thing in common. They all imply that your computer will do more than one thing at a time. Well the Tandy 200 is a computer, right? So if it could do two things at the same time, why not put that ability to some good use? My needs, as a writer and a programmer, were to be able to use the Tandy 200 while printing long copy, or program listings.

Because this article is about customizing your 200, let's fix *F8* in the Menu to print to *LPT:* the *.DO* file that is under the widebar cursor. If the cursor is not over a *.DO* file, then no action is taken. A beep will announce the beginning and end of the printout. In the program in this article, Line-feeds are not sent to the printer, but you can modify the code to suit your needs.

BE CAREFUL

If a file is printing, you can abort it by pressing *F8* again at the main Menu. You must be careful not to use any print commands while *background printing* is operating. Background printing is the technique of having the computer do other jobs while it is also printing a file.

By including the additional code for background printing we have exceeded the 256 bytes that were protected by our LOMEM installation in the previous two articles. We must now set N=2 (in line 2 of *LOMEM.DO*) and reserve 512 bytes.

To adjust *LOMEM* we must start with a completely empty bank. Cold starting a RAM bank is the quickest way to accomplish this task. Next we must load, copy, or type in the following *TEXT* file.

This must be a *TEXT* file, NOT a *BASIC* program, and this *TEXT* file must be called *LOMEM.DO*. The variable N in line 2 is set for the number of 256 byte blocks you wish to reserve.

With *LOMEM.DO* as the only file

```
1 Kill "LOMEM.DO"
2 N=2:AD=40960
3 POKE AD,N
4 POKE AD+256*N,O
5 POKE 62703,160+N
6 NEW
```

Listing 1. LOMEM.DO

in the RAM bank, enter *BASIC* and *RUN "LOMEM"*. This program will end with a *SYNTAX* error but that is correct. At the *OK* prompt type *NEW* and press *ENTER*> (this is very important). Now there should be NO syntax error. When you return to the Menu *LOMEM.DO* will no longer be there and the Bytes Free message will be 19078 (19590 - 512). We now have a place to store and execute the code for our keyscan *hook*.

SOME PROGRAMMING TECHNIQUES

You may have noticed that my code in listing 3 does not use additional RAM for variable storage or counters. The byte where data is stored is frequently part of the instruction needing that data. Notice in the *PRNT* routine, the storage area for the *DE* and *BC* registers is located at *PRNT+1* and *BCNT+1*, and similarly *P.FLAG* is located at *PFLAG+1*. This saves bytes and execution time.

Another programming technique that should be mentioned is the use of the offset X. This variable is set in the first *EQU* statement. The code for *LOMEM HOOK CODE* can be assembled at any location and relocated to any location with the formula: *X: EQU (org adrs +3) - resident address*

That's why all internal jumps and calls have "*-X*" after them. This allows for the relocation of the code by the *START:* routine.

Printing a *.DO* file via *F8* is the same as saving a *.DO* file to *LPT:*. Embedded control characters and escape sequences are sent and will control your printer. Output from a word processor, like Write ROM or T-WORD, can be sent to a RAM file and then printed from the Menu. This will allow a formatted printout while perhaps editing another document. I personally like to log onto a local BBS while waiting for a long

printout to complete.

I usually have a file called *EMPH.DO* on the menu. This is only 4 bytes and contains *ESC E CRLF*. By placing the cursor over this file and pressing *F8*, the printer is put into the *emphasized* mode. This could be done for Compressed, Italics, or whatever mode you use often. After all, not all of your *.DO* files will have print commands embedded within them.

Note: due to the dynamic relocation of RAM files during certain Tandy 200 operation, background printing will be suspended at times. Printing will resume when all appropriate internal *pointers* are properly re-adjusted.

Please notice in the *HOOK* routine the *CALL PBYTE* occurs *BEFORE* we check to see where we are. This will allow the *PBYTE* routine to send a character to the printer with every keyscan. When at the Menu, the remaining unused function keys will be tested and provisions for expansion have been included. *continued*


```
0 CLS:CLEAR256,60000
1 FOR I=60000 TO 60424
2 PRINT@50,I:READ X:POKE I,X:SM=SM+X
3 NEXT
4 IF SM=50431 THEN CALL 60000
5 BEEP:PRINT"error in data":STOP
100 DATA 195,217,235,42,13,245,17,61,160
101 DATA 223,202,164,103,34,123,160,235
102 DATA 34,13,245,195,164,103,66,97,110
103 DATA 107,32,66,97,99,107,32,32,32,32
104 DATA 32,32,67,111,112,121,32,75,105
105 DATA 108,108,32,32,32,32,32,32,70
106 DATA 105,108,101,32,80,114,110,116,0
107 DATA 245,197,213,229,62,0,167,196
108 DATA 140,160,56,10,237,17,188,104
109 DATA 223,194,118,160,205,150,79,33
110 DATA 21,160,205,204,17,58,12,253,254
111 DATA 128,202,191,160,254,64,202,250
112 DATA 160,254,32,202,118,160,254,4
113 DATA 202,118,160,254,2,202,103,161
114 DATA 225,209,193,241,195,168,156,58
115 DATA 72,248,135,95,22,0,33,201,247
116 DATA 25,235,237,126,201,219,187,230
117 DATA 4,192,17,0,0,237,229,14,8,17
118 DATA 242,160,205,17,110,225,192,1,0
119 DATA 0,9,126,254,26,194,179,160,175
120 DATA 50,66,160,205,69,79,62,13,254
121 DATA 10,196,201,132,3,96,105,34.161
122 DATA 160,201,58,66,160,167,194,236
123 DATA 160,205,125,160,254,192,194,118
124 DATA 160,205,69,79,50,66,160,35,34
125 DATA 146,160,6,8,235,237,235,33,242
126 DATA 160,205,186,65,33,0,0,34,161
127 DATA 160,195,118,160,205,170,160,195
128 DATA 118,160,48,48,48,48,48,48,48,48
129 DATA 205,77,79,6,0,205,22,161,6,4
130 DATA 205,22,161,6,8,205,22,161,175
131 DATA 50,30,253,205,247,18,195,164
132 DATA 103,33,181,242,205,30,0,205,98
133 DATA 161,254,255,204,62,79,202,62,79
134 DATA 230,128,194,51,161,17,11,0,25
135 DATA 195,28,161,35,35,35,205,98,161
136 DATA 254,32,202,63,161,231,62,5,167
137 DATA 202,76,161,61,50,64,161,195,53
138 DATA 161,62,46,231,35,205,98,161,231
139 DATA 35,205,98,161,231,62,5,50,64
140 DATA 161,35,195,25,161,205,177,155
141 DATA 122,201,243,219,216,198,4,230
142 DATA 12,254,12,202,106,161,71,195
143 DATA 105,155,1,118,1,33,99,234,17,1
144 DATA 160,205,22,131,6,22,17,243,235
145 DATA 33,69,241,205,186,65,195,1,160
146 DATA 72,111,111,107,24,67,97,108,108
147 DATA 52,48,57,54,49,13,0,69,100,105
148 DATA 116,32,0,
149 REM          END OF DATA
```

Listing 1. The BASIC code for giving your Tandy 200 reverse bank switching, free space list for all banks, and background printing.

```
;*********************************
;* LOMEM HOOK CODE BY P.GLOBMAN *
;*      Copyright (C) 1987       *
;*********************************
;
X:      EQU 60003-A001H ;CALC OFFSET
BEEP:   EQU 4F45H
CLS:    EQU 4F4DH
CRLF:   EQU 4F3EH
F_SLOT: EQU 63560
GONE:   EQU 9CA8H
HOOK04: EQU F50DH
MENU:   EQU 67A4H
POINT:  EQU 4F96H
PRINT:  EQU 11CCH
TABLE:  EQU F7C9H
WAIT:   EQU 12F7H
;----------------
        ORG 60000
        JMP START       ;TO RELOCATOR
;
LOADER: LHLD HOOK04     ;ORIG. JUMP
        LXI D,HOOK-X    ;ADRS OUR HOOK
        RST 3           ;CMP HL - DE
        JZ MENU         ;ALREADY THERE
;
        SHLD BYE+1-X    ;BYE+1=OLD HOOK
        XCHG
        SHLD HOOK04     ;HOOK=OUR CODE
        JMP MENU
;=================================
LABEL:  DB 'Bank Back      Copy '
        DB 'Kill       File Prnt',0
;----------------
HOOK:   PUSH PSW        ;SAVE ALL REG
        PUSH B
        PUSH D
        PUSH H
;
;
PFLAG:  MVI A,0         ;P.FLAG=PFLAG+1
        ANA A           ;PRINTING?
        CNZ PBYTE-X     ;IF YES CALL
;
        DESP 10         ;LOOK AT SP 10
        LHLI            ; BYTES BACK
        LXI D,68BCH
        RST 3           ;CMP TO 68BCH
        JNZ DONE-X      ;IF EQ->AT MENU
;
;AT MAIN MENU
;
        CALL POINT      ;PRINT LABEL
        LXI H,LABEL-X   ; LINE ON THE
        CALL PRINT      ;  MAIN MENU
;
        LDA 64780       ;CHECK LAST KEY
        CPI 128
        JZ PRT_IT-X     ;IF F8 -> PRNT
        CPI 64
        JZ S_FILE-X     ;IF F7 -> S_FIL
;
        CPI 32
        JZ DONE-X       ;F6 AND F3 ARE
        CPI 4           ; RESERVED FOR
        JZ DONE-X       ;FUTURE MODS
;
        CPI 2
        JZ BNK-X        ;IF F2 -> BACK
;
DONE:   POP H           ;ELSE RESTORE
        POP D           ;AND EXIT
        POP B           ;BYE+1 (GONE)
        POP PSW         ;is altered by
BYE:    JMP GONE        ;loader subrtne
;=================================
GET:    LDA F_SLOT      ;GET FILE
        ADD A           ;ADJ OFFSET AND
        MOV E,A         ;PUT IN DE REG
        MVI D,0         ;THEN ADD IT TO
        LXI H,TABLE     ;ADRS OF TABLE
        DAD D           ;HL=ADRS OF DIR
        XCHG            ;GET THAT ADRS
        LHLI            ;TO HL AND MOVE
        MOV A,M         ;ATT BYTE ->A
        RET
;=================================
;Print in background (called by hook)
;
PBYTE:  IN BBH          ;LOOK AT LPT:
        ANI 04          ;BIT 3=NOT RDY
        RNZ             ;IF NR THEN RET
PRNT:   LXI D,00H       ;DE=PRNT+1
        LHLI            ;location (top)
;
        push h
        MVI C,8         ;GET LEN TO C
        LXI D,SIGN-X    ;SRCH FOR SIGN
        CALL 6E11H      ;COMPARE STRNGS
        pop h
        RNZ             ;file not there
;
BCNT:   LXI B,00H       ;BC=BCNT+1
        DAD B
        MOV A,M         ;GET THE BYTE
        CPI 26          ;IF NOT EOF
        JNZ P_CHAR-X    ; THEN PRINT IT
;
EOF:    XRA A           ;ELSE
        STA PFLAG+1-X   ; RESET P.FLAG
        CALL BEEP       ;ANNOUNCE END
        MVI A,13        ;  AND LPRINT
;
P_CHAR: CPI 10          ;IF NOT LF THEN
        CNZ 84C9H       ;PRINT THIS CHR
        INX B           ;POINT TO NEXT
        MOV H,B         ;SAVE BYTE CNTR
        MOV L,C         ;IN 'BC'
        SHLD BCNT+1-X   ;
        RET             ;AND RETURN
;=================================
;Print in background (init by F-key)
;
PRT_IT: LDA PFLAG+1-X   ;PRINTING? LOOK
        ANA A           ;AT P.FLAG
        JNZ BUSY-X      ;(PBYTE+1)
        CALL GET-X      ;LOOK FOR .DO
        CPI 192         ;FILE. JMP DONE
        JNZ DONE-X      ;IF NO .DO FILE
;
DOFILE: CALL BEEP       ;ANNOUNCE START
        STA PFLAG+1-X   ;SET P.FLAG
        INX H           ;FILE START ADR
        SHLD PRNT+1-X   ;IN 'DE'
;
        MVI B,8         ;Record 8 chrs
        XCHG            ;as signature
        LHLI
        XCHG
        LXI H,SIGN-X    ;dest
        CALL 41BAH      ;move it
;
        LXI H,0000H     ;BYTE COUNTER
        SHLD BCNT+1-X   ;IN BC
        JMP DONE-X
;
BUSY:   CALL EOF-X
        JMP DONE-X
;
SIGN:   DB '00000000'
;=================================
S_FILE: CALL CLS        ;SHOW ALL FILES
        MVI B,0         ;FIRST BANK
        CALL BANK-X
        MVI B,4         ;SECOND BANK
        CALL BANK-X
        MVI B,8         ;THIRD BANK
        CALL BANK-X
        XRA A
        STA 64798       ;0 keystrokes
        CALL WAIT
        JMP MENU
;
BANK:   LXI H,F2B5H     ;USER FILE #1
S_LOOP: CALL 1EH        ;PRINT SPACE
S1:     CALL GET_B-X    ;ATTRIB BYTE
        CPI FFH         ;NO MORE FILES
        CZ CRLF         ;THEN RETURN
        JZ CRLF
        ANI 80H         ;ACTIVE FILE?
        JNZ FILE-X      ;YES-SHOW IT!
        LXI D,11        ;NO-GET NEXT
        DAD D           ;FILE SLOT AND
        JMP S1-X        ;DO IT AGAIN!
;
FILE:   INX H           ;SKIP ATTRIB
        INX H           ;BYTE AND ADRSS
C_LOOP: INX H
        CALL GET_B-X    ;GET FILENAME
        CPI ' '         ;SKIP SPACES
        JZ SKP-X
        RST 4           ;PRINT FILENAME
SKP:    MVI A,5         ;ANY MORE CHRS?
        ANA A
        JZ C_DONE-X     ;NO-THEN DONE
        DCR A           ;ADJUST COUNTER
        STA SKP+1-X     ;STORE IT AND
        JMP C_LOOP-X    ;GET NEXT CHR
;
C_DONE: MVI A,'.'       ;PRINT THE DOT
        RST 4
        INX H           ;AND FILE EXT
        CALL GET_B-X
        RST 4
        INX H
        CALL GET_B-X
        RST 4
        MVI A,5         ;RESTORE COUNT
        STA SKP+1-X
        INX H           ;AND DO NEXT
        JMP S_LOOP-X    ;FILE SLOT
;
GET_B:  CALL 9BB1H      ;MAKE THIS PEEK
        MOV A,D         ;PUT BYTE IN A
        RET
;=================================
;REVERSE BANK SWITCHING
;
BNK:    DI              ;INSTEAD OF
        IN D8H          ;BACK ONE BANK,
B1:     ADI 4           ;WERE REALLY
        ANI 0CH         ;GOING AHEAD
        CPI 0CH         ;TWO BANKS BY
        JZ B1-X         ;MAKING THE ROM
        MOV B,A         ;THINK WE BEGAN
        JMP 9B69H       ;1 BANK AHEAD
;=================================
;MOVE HOOK INTO PLACE AND INITIALIZE
;
START:  LXI B,START-LOADER ;# of bytes
        LXI H,LOADER    ;source
        LXI D,LOADER-X  ;destination
        CALL 8316H      ;move block
;
;FIX BASIC F6 / F7
;
        MVI B,22        ;# of bytes
        LXI D,KEY_6     ;source
        LXI H,F145H     ;destination
        CALL 41BAH      ;move it
;
        JMP LOADER-X    ;INITIALIZE
;
KEY_6:  DB 'Hook',24,'Call40961',13,0
        DB 'Edit ',0
```

Listing 1. The assembly code for giving your Tandy 200 reverse bank switching, free space list for all banks, and background printing.

FOR BASIC PROGRAMMERS

The *BASIC* program in listing 2 will poke the code into memory, maintain a checksum for data accuracy, and execute the code if all is correct. Whether you use the *BASIC* program or the assembly source code, be sure to run *LOMEM.DO* first.

The program listings (*BASIC* and *ASM*) include the code for stripping *LF*'s. If you need to send *LF*'s to your printer, you may do the following:

In the *ASM* file, at the *P_CHAR:* routine, replace *CPI 10* with *CPI 0*.

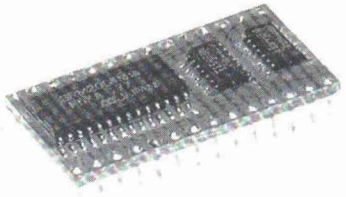
In the *BASIC* program, at line #121, replace the *10* with a *0*, and change line #4 to *4 IF SM=50421 THEN CALL 60000*

I would like to extend a note of thanks to James Yi (73327,1653) for beta testing this project and providing valuable input towards its completion.

Next month I will discuss the *chaining* or *linking* of programs in different RAM banks.

--by Paul Globman

EXPANSIONS!!!

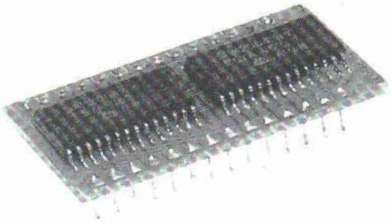


MODEL 100/NEC PC-8201A
8K Memory Module
$19 each

Easy to install. Open the case and plug-them-in. Each module expands your memory by 8K bytes. Extra low power components mean long battery life. Very low profile means a proper fit in the NEC PC-8201A. Detailed instructions make installation quick and easy. You can expand your Model 100 to 32K and your PC-8201 to 64K (2 banks of 32K each).

NEW!!!
MODEL 102
8K Memory Module
only $9.95

Easy to install. Just open the hatch and plug it in.



TANDY 200
24K Memory Module
$49 each

Simply pop open the hatch with a coin and plug in one or two of these modules. Each module adds a 24K bank of memory to the TANDY 200. It's like getting two more machines. The built in COPY function key copies files between banks instantly. Like our 8K, we build these with the lowest power and most reliable memory chips available.

THE PURPLE POLICY

Try any of our products for 30 days, satisfy yourself that our service, quality, and prices add up to the best value anywhere—if not completely satisfied, you can return the product for a full refund. Prices include UPS surface shipping (in Continental USA)—even the phone call is free.

30-day money-back guarantee.

IT'S EASY TO ORDER

Send your order with payment to the address below. Or, if you prefer, credit card orders can be handled by phone—VISA, MasterCard, and American Express are welcome. California residents add 6% sales tax. Checks allow 3 weeks to clear.

1-800-732-5012 TOLL FREE
Orders only (8am—5pm PST)

(805) 987-4788 In California
For orders or customer service

PURPLE COMPUTING

420 Constitution Ave.
Camarillo, CA 93010

Telex: 888661 (PURPLE)

Canada: Canada Portable Computer, (604) 534-6441
Australia: Softech Computer Services, (2) 419-8899

Circle 74 on reader service card

Not Sure About A Product Or Service You've Seen Here?

Then why not fill out the Reader Service Card right now, send it to us, and you'll receive more information directly from the vendor or manufacturer.

Data Acquisition System For the Model 100/102 and Model 4

Provides engineers, technicians, educators, students and experimenters with a flexible solution to data acquisition and interface needs using the Model 102 and Model 4 computers:

M102-DACQ-1 . . . . . **$185.**
8 digital input lines; 8 digital output lines; 8 channel, 8 bit A/D (ADC0809) 8 bit D/A (DAC0831); 3 channel, 16 bit prog timer (82C53); auto on/off; interfaces to system bus; 3 sq. in. wire wrap; user manual, schematics, programming instructions/examples. Board (4.2″ x 7.6″) cabling and AC adapter. No case.

M102-DACQ-TERM . . . . . **$100.**
Terminal board for DACQ card and customer hardware; all DACQ signals available; screw terminals and diode clamps for all analog inputs; 12 sq. in. wire wrap; stand-offs to mount above M102-DACQ. Cabling provided.

HI CAPACITY BATTERY PACK . . . . . **$95.**
Sealed lead acid; outputs for both M102 and DACQ; charger, case.

Systems

M102-DACQ SYS1 . . . . . **$295.**
Both DACQ-1 and DACQ-TERM cards mounted in case with terminal card exposed for easy access.

M102-DACQ SYS2 . . . . . **$320.**
Both DACQ-1 and DACQ-TERM cards enclosed in instrument case. Cut-outs for cabling provided.

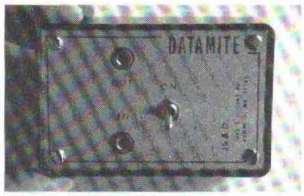
Specify M100-DACQ for Model 100 interface cables
Specify M4-DACQ for Model 4 interface cables.

Postage Paid, M/C, VISA, money order, check, approved P.O.
Sales Tax: add 4% (VT residents)

Rural Engineering Inc.

Route 14, Box 113F, So. Royalton, VT 05068 Tel. (802) 763-8367

Circle 88 on reader service card



THE DATAMITE

LOW COST ANALOG TO DIGITAL CONVERSION FOR YOUR TANDY LAPTOP

Plug the cassette input of your Tandy 100/102/200 into the DATAMITE and it will provide high resolution, ±.04% accurate A/D conversion over a 0-5 volt input range. Companion chart recorder software samples input voltage of DATAMITE at user-selected intervals, records value in RAM, and plots a calibrated scrolling graphic display of voltage data on your computer screen. See August 1987 issue of *Portable 100* for a complete description of the DATAMITE. (NEC,Olivetti versions also.)

DATAMITE	$45.00	**Jones Service & Design**
Chart Recorder Software	$24.00	1842 S. Nugent Road
DATAMITE Kit	$35.00	Lummi Island, WA 98262
VISA/Master Card		(206) 758-7258

The Portable 100 Classifieds

HOW TO PLACE A CLASSIFIED AD:

Categorize your advertisement (Hardware, Software, Services, Wanted, Etc.) and carefully type your message. We are not responsible for correctly interpreting handwritten advertisements. Phone numbers, street numbers, dimensions, and any abbreviations count as one word. Logos, company or product, are not allowed, neither are display advertisements. Business rates are $1.60 per word; non-business rates (individuals advertising) are $1.30 per word. Add up the cost and send the advertisement copy with a check, money order, Visa, or Mastercard number to: Portable 100 Classifieds, 145 Grove St. Ext., P.O. Box 428, Peterborough, NH 03458-0428, c/o Linda Tiernan. Make checks payable to Portable 100 Classifieds. Include your complete return address and phone number (phone number is printed only if it is included in the advertisement itself). Materials due the 1st of the month, two months prior to the magazine cover date (example: materials for the February issue must be received by December 1st.) Advertisements received after the deadline will appear in the next scheduled issue. Payment must accompany order. No refunds for advertisements that miss deadlines, regardless of reason. We reserve the right to change advertisement categories, and to reject, edit, or cancel any advertisement we deem improper. There are no agency discounts for classifieds. For faster service call 603-924 7949.

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

Ad Category_____ No.of words x (1.60)(1.30) _____ #Issues to run __

Name_____

Address _____

City _____ State _____ Zip _____

Total Enclosed x (Ad Cost x Issues) _____ Phone Number _____

Visa/Mastercard Number_____ Exp. Date _____

Signature _____

If more space is needed, please use a separate sheet of paper.

Free Information

For free information on products advertised in this issue of Portable 100, locate the Reader Service number corresponding to the advertisement that interests you. Circle the number on the Reader Service Card at the center of the magazine (or on the wrapper protecting the magazine if you are a subscriber) and drop it into the mail. The literature you've requested will be forwarded to you without any obligation. Please allow 3--5 weeks for delivery.

ADVERTISER

** Please contact this advertiser directly

Don't forget to say you saw it in Portable 100!

The sky's the limit...
...with Portable 100 magazine.

On the road . . . at home . . . in the office. . . in the air. . . the variety of tasks you can perform with your Tandy portable is expanding constantly.

To keep fully informed and up-to-date about current trends, new products, and new uses for your laptop, you need **Portable 100** magazine.

From sophisticated input/output (I/O) calls and their applications, to simple disk drives, **Portable 100** magazine covers it all.

Portable 100 gives you features, news, columns, and reviews that are thorough and timely. And they are written in a fast-paced, easy-to-read style, by leading experts in the computer field.

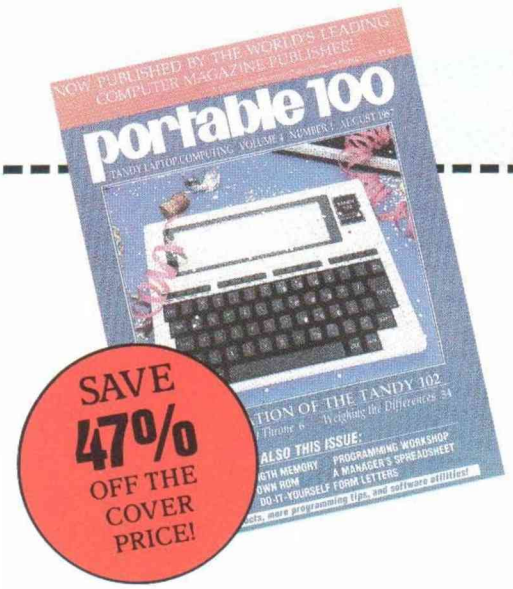
In upcoming issues you will discover. . .

- how new peripherals can make you more productive.
- more efficient ways to communicate with your desktop.
- where to buy low-cost public-domain software.
- how your business can be more profitable with your portable at your fingertips.
- where to find the best bulletin boards and information services.
- and much more!

Don't miss even a single valuable monthly issue. Fill out the coupon below, or to charge it to your credit card, call

1-603-924-7949

Send no money now! We will gladly bill you.



☐ **YES!** I want to explore ways to be more productive with my Tandy portable. . . and save 47% off the newsstand price. Send me a year's subscription (12 issues) of **Portable 100** for $24.97.

☐ Payment enclosed ☐ Bill me
Make checks payable to **Portable 100**

Name _____

Address _____

City _____ State ____ Zip _____

Canada $45.97 (Canadian funds), Mexico, $29.97. Foreign Surface $44.97. One year only. U.S. Funds drawn on U.S. banks. Foreign Airmail, please inquire. Please allow 6-8 weeks for delivery.

37DB4

MONEY BACK GUARANTEE:
If you are not completely satisfied with **Portable 100,** you may cancel your subscription and receive a full refund. Please allow 6-8 weeks for delivery of your first issue.

SUPER ROM

Lucid Spreadsheet Write ROM Database Outliner

NOW YOU CAN REALLY HAVE IT ALL!

All on one ROM. Truly the finest four programs available for the Model 100 — guaranteed. Try it for 30 days. If you aren't blown away by the excellence return it for a full refund.

$199.95

The four best programs for the Model 100 all on one ROM. 32K of power without using any RAM for program storage. This is the PCSG Snap-In ROM that just presses easily into the little ROM socket in the compartment on the back. You access the four right from the main menu like built-ins.

Write ROM — the definitive word processor for the Model 100. Function key formatting or dot commands. Search and replace. Library feature — inserts words, phrases or whole documents into text from just a code. MAP lets you see a picture of your document. In all there are 60 features and functions. No one can claim faster operation. FORM lets you create interactive forms with on-screen prompts that you can answer from the keyboard. Nothing else for the Model 100 compares with the features of Write ROM. Exactly the same as the Write ROM sold as a single program. Infoworld says it "makes the Model 100 a viable writing unit ... surpassed our highest expectations for quality and clarity."

Lucid Spreadsheet: This is the one PICO magazine says "blows Multiplan right out of the socket" and Infoworld performance rated as "excellent" and said "makes the Model 100 compute." Gives you features you cannot get with Lotus 123. Lets you build spreadsheets in your Model 100 that would consume 140-150K on a desktop. Program generating capability with no programming knowledge required. Variable column widths. Includes find and sort with function key control. It's fast, recalculates like lightning. No feature has been taken from the original, only new ones added.

Database: This is a relational data base like no other. You can do everything from mailing lists to invoices. No complicated pseudo-coding, you create input screens as simply as typing into TEXT. You are not limited by size; you can have as large an input screen as you wish. Prints out reports or forms, getting information from as many files as you like. Complete math between fields. Total interface with Lucid worksheets.

Outliner: Does everything that Thinktank does on a PC but a whole lot better. Includes a Sort for your headlines. Lets you have headlines of up to 240 characters. Has cloning, hoisting and sideways scroll up to 250 characters. Like Lucid, this one sets a new standard for outliners. This is the way to plan and organize your projects.

Present Lucid and Write ROM owners can upgrade for $150. If you have both it's $125.

As usual PCSG sells the Super ROM on a thirty day guarantee. If for any reason you are not satisfied, simply return it for a full refund.

We are excited about this product. Super ROM gives the Model 100 the true power of a desktop. No other multi-program ROM has software that compares. But don't take our word for it. We invite you to make that comparison yourself. Priced at $199.95 on Snap-In ROM.

Got stuck with somebody else's multi-ROM? We'll upgrade it for $150.

(214) 351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd., #206, Dallas, TX 75229

MC, Visa, American Express, Check, or C.O.D.

Circle 40 on Reader Service card.